Combining the Burrows Wheeler Transform and the Context-Tree Weighting Algorithm

Frans M.J. Willems

Department of Electrical Engineering
Eindhoven University of Technology

1st IEEE Seminar on Future Directions in & 2nd African Winter School on Information Theory and Communications, August 16-21, 2015, Protea Kruger Gate, South Africa
Motivation

European School on Information Theory, April 20-24, 2015, Zandvoort, The Netherlands:

Tutorial lectures were given by Young-Han Kim, Stephanie Wehner, Imre Csiszar, and Stephan ten Brink and
Motivation

Richard Durbin (Genome Informatics Group, Sanger Institute, Cambridge, UK)

Human Genome Project, now 99598 citations, h-index = 102, now co-leader of 1000 Genomes Project.

“Storage and Search of Genome Sequence Information”.

CLASSES of UNIVERSAL SOURCE CODING ALGORITHMS:

- **String Matching methods**

- **Source-Modeling and Arithmetic Coding methods**
  Context-Tree Weighting (CTW) Method

- **Burrows-Wheeler (BW) transform, followed by Move-to-Front and Huffman Coding Methods**
  BW: 1994
**Problem Description**

**Computer Science** focusses on **minimizing Memory and Computational Complexity.**

**Information Theory** focusses on **minimizing Redundancy.**

BW is very efficient, allows for searching in the compressed domain, but is **SUBOPTIMAL in terms of redundancy** for tree sources. BW does NOT achieve the Rissanen lower bound ($1/2 \log_2 N$ bits per parameter, 1984, **IT-Soc Best Paper Award**).

Effros-Visweswariah-Kulkarni-Verdu (2002): in the binary case $\log_2 N$ bits per parameter **extra redundancy** per parameter.

Fixed-depth CTW is linear in $N$ (memory and computations) but is **REDUNDANCY OPTIMAL** in Rissanen sense ($1/2 \log_2 N$ bits per parameter) for tree sources. CTW does not allow for searching in the compressed domain however.

**PROBLEM:**

Improve the redundancy of BW data-compression.
Computer Science focusses on minimizing Memory and Computational Complexity.

Information Theory focusses on minimizing Redundancy.

BW is very efficient, allows for searching in the compressed domain, but is **SUBOPTIMAL in terms of redundancy** for tree sources. BW does NOT achieve the Rissanen lower bound ($1/2 \log_2 N$ bits per parameter, 1984, IT-Soc Best Paper Award).

Effros-Visweswariah-Kulkarni-Verdu (2002): in the binary case $\log_2 N$ bits per parameter *extra redundancy* per parameter.

Fixed-depth CTW is linear in $N$ (memory and computations) but is **REDUNDANCY OPTIMAL** in Rissanen sense ($1/2 \log_2 N$ bits per parameter) for tree sources. CTW does not allow for searching in the compressed domain however.

**PROBLEM:**

Improve the redundancy of BW data-compression.
INTRODUCTION
IID SOURCES, PREFIX CODES, REDUNDANCY
ENUMERATIVE CODING
ARITHMETIC CODING
CONTEXT-TREE WEIGHTING
BURROWS WHEELER
CODING BW-SEQUENCES
CODING B-COUNTS
FINDING BEST TREE MODEL
BINARY DECOMPOSITION
CONCLUSION
FUTURE DIRECTIONS
The binary source produces a sequence $x_1^N = x_1 x_2 \cdots x_N$ with components $\in \{0, 1\}$ with probability $P(x_1^N)$.

**Definition (Binary IID Source)**

For an independent identically distributed (i.i.d.) source with parameter $\theta$, for $0 \leq \theta \leq 1$,

$$P(x_1^N) = \prod_{n=1}^{N} P(x_n),$$

where $P(1) = \theta$, and $P(0) = 1 - \theta$.

A sequence $x_1^N$ containing $w$ ones (i.e., having weight $w$) and $N - w$ zeros has probability

$$P(x_1^N) = (1 - \theta)^{N-w} \theta^w.$$

**Definition (Entropy Binary IID Source)**

The entropy of this source is $h(\theta) \triangleq (1 - \theta) \log_2 \frac{1}{1 - \theta} + \theta \log_2 \frac{1}{\theta}$ (bits).
Prefix Codes

**Definition (Prefix Code)**

In a **prefix code** no codeword is the prefix of any other codeword.

We restrict ourselves to prefix codes. Codewords in a prefix code can be regarded as **leaves in a rooted tree**. Prefix codes lead to **instantaneous decodability**.

**Example**

<table>
<thead>
<tr>
<th>$x_1^N$</th>
<th>$c(x_1^N)$</th>
<th>$L(x_1^N)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>01</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>110</td>
<td>3</td>
</tr>
<tr>
<td>11</td>
<td>111</td>
<td>3</td>
</tr>
</tbody>
</table>

![Tree diagram for prefix codes](image)
Theorem (Kraft, 1949)

(a) The lengths of the codewords in a prefix code satisfy Kraft's inequality

\[ \sum_{x_1^N \in \mathcal{X}^N} 2^{-L(x_1^N)} \leq 1. \]

(b) For codeword lengths satisfying Kraft's inequality there exists a prefix code with these lengths.
This leads to:

**Theorem (Fano, 1961)**

(a) Any prefix code satisfies

\[ E[L(X_1^N)] \geq H(X_1^N) = Nh(\theta). \]

*The minimum is achieved if and only if* \( L(x_1^N) = \log_2 \left( \frac{1}{P(x_1^N)} \right) \) (ideal codeword length) *for all* \( x_1^N \in X^N \) *with nonzero* \( P(x_1^N) \).

(b) There exist prefix codes with \( L(x_1^N) = \left\lceil \log_2 \left( \frac{1}{P(x_1^N)} \right) \right\rceil \), hence

\[ L(x_1^N) < \log_2 \frac{1}{P(x_1^N)} + 1 \]

(ideal codeword length plus 1 bit). *These codes achieve*

\[ E[L(X_1^N)] < H(X_1^N) + 1 = Nh(\theta) + 1 \text{ bit}. \]
Redundancy

Definition

The **individual redundancy** $\rho(x_1^N)$ of a sequence $x_1^N$ is defined as

$$\rho(x_1^N) = L(x_1^N) - \log_2 \frac{1}{P(x_1^N)},$$

i.e., **codeword-length minus ideal codeword-length**.

Definition

The **expected redundancy** $\rho(x_1^N)$ is defined as

$$\rho = E[L(X_1^N)] - E[\log_2 \frac{1}{P(X_1^N)}] = E[L(X_1^N)] - H(X_1^N),$$

i.e., **expected codeword-length minus entropy**.

Note that there exist codes with individual redundancies ≤ 1 and consequently with expected redundancies ≤ 1.
Indexing Sequences of Fixed Weight

**IDEA:**

Sequences having the same number of ones (weight) have the same probability and only need to be INDEXED.

**Definition (Lexicographical Ordering)**

In a lexicographical ordering $(0 < 1)$ we say that $x_1^n < y_1^n$ if $x_n < y_n$ for the smallest index $n$ such that $x_n \neq y_n$.

Consider a subset $S$ of the set $\{0, 1\}^N$. Let $i_S(x_1^n)$ be the lexicographical index of $x_1^n \in S$, i.e., the number of sequences $y_1^n < x_1^n$ for $y_1^n \in S$.

**Example**

Let $N = 5$ and $S = \{x_1^5 : w(x_1^5) = 3\}$ where $w(x_1^5)$ is the weight of $x_1^5$. Then $|S| = \binom{5}{3} = 10$ and:

- $i_S(11100) = 9$
- $i_S(10011) = 4$
- $i_S(11010) = 8$
- $i_S(01110) = 3$
- $i_S(11001) = 7$
- $i_S(01101) = 2$
- $i_S(10110) = 6$
- $i_S(01011) = 1$
- $i_S(10101) = 5$
- $i_S(00111) = 0$
Let $N = 5$ and $S = \{x_1^N : \sum x_n = 3\}$. Then $|S| = \binom{5}{3} = 10$.

\[ i(11010) = 4 + 3 + 1 = 8. \]
Example (From Index to Sequence)

Again \( N = 5 \) and \( S = \{x_1^N : \sum x_n = 3\} \).

Index \( i = 8 \), now (a) \( 8 \geq 4 \) hence \( x_1 = 1 \), (b) \( 8 \geq 4 + 3 \) hence \( x_2 = 1 \), (c) \( 8 < 4 + 3 + 2 \) hence \( x_3 = 0 \), (d) \( 8 \geq 4 + 3 + 1 \) hence \( x_4 = 1 \), (e) \( x_5 = 0 \).
If sequence $x_1^N$ has weight $w$, the index $i(x_1^N)$ is encoded using a fixed-length code of length

$$L_i(x_1^N) = \left\lceil \log_2 \left( \frac{N}{w} \right) \right\rceil.$$ 

**Example**

Index $i_S(11010) = 8$. Since $|S| = 10$ the length of the fixed-length code for $i(x_1^N)$ is 4, and the corresponding codeword could be 1000.
How to Code the Weight? $\theta$ Coding

The weights that can occur are $w \in \{0, 1, \cdots, N\}$. Knowing source parameter $\theta$ we can easily compute the probability that weight $w$ occurs

$$P_\theta(w) = \binom{N}{w} (1 - \theta)^{N-w} \theta^w.$$ 

The weight $w$ of sequence $x_1^N$ can therefore be encoded with a variable-length code of length

$$L_\theta(w) = \left\lceil \log_2 \frac{1}{\binom{N}{w} (1 - \theta)^{N-w} \theta^w} \right\rceil.$$ 

Now we can write for the total codeword length for sequence $x_1^N$ having weight $w$

$$L(x_1^N) = L_\theta(w) + L_i(x_1^N)$$

$$= \left\lceil \log_2 \frac{1}{\binom{N}{w} (1 - \theta)^{N-w} \theta^w} \right\rceil + \left\lceil \log_2 \binom{N}{w} \right\rceil$$

$$\leq \log_2 \frac{1}{(1 - \theta)^{N-w} \theta^w} + 2.$$
\( \theta \) Coding: Individual and Expected Redundancy

Assume that our sequence \( x_1^N \) has weight \( w = w(x_1^N) \), then for all \( 0 \leq \theta \leq 1 \), we get for the individual redundancy

\[
\rho(x_1^N) = L(x_1^N) - \log_2 \frac{1}{(1 - \theta)^{N-w} \theta^w} 
\leq \log_2 \frac{1}{(1 - \theta)^{N-w} \theta^w} + 2 - \log_2 \frac{1}{(1 - \theta)^{N-w} \theta^w} = 2 \text{ bits.}
\]

For the expected redundancy, when \( 0 \leq \theta \leq 1 \), we get

\[
\rho = E[L(X_1^N)] - Nh(\theta) \leq 2 \text{ bits.}
\]
How to Code the Weight? Uniform Coding

The weights that can occur are \( w \in \{0, 1, \cdots, N + 1\} \), hence there are \( N + 1 \) alternatives. Therefore we can use a \textbf{uniform code} of length

\[
L_U(w) = \lceil \log_2(N + 1) \rceil.
\]

Now we can write for the total codeword length for sequence \( x_1^N \) having weight \( w \)

\[
L(x_1^N) = L_U(w) + L_I(x_1^N)
\]

\[
= \lceil \log_2(N + 1) \rceil + \left\lceil \log_2\left(\binom{N}{w}\right) \right\rceil
\]

\[
\leq \log_2(N + 1) \binom{N}{w} + 2.
\]
Uniform Coding: Individual Redundancy

Assume that our sequence $x_1^N$ has weight $w$, then for all $0 \leq \theta \leq 1$, then we get for the individual redundancy

$$\rho(x_1^N) = L(x_1^N) - \log_2 \frac{1}{(1 - \theta)^{N-w\theta}}$$

$$\leq \log_2 (N + 1) \binom{N}{w} + 2 - \log_2 \frac{1}{(1 - \theta)^{N-w\theta}}.$$

If both $N - w \to \infty$ and $w \to \infty$ the Stirling approximation yields

$$(N + 1) \binom{N}{w} \approx (N + 1) \frac{\sqrt{2\pi N}}{\sqrt{2\pi (N - w)} \sqrt{2\pi w}} \left( \frac{N}{N - w} \right)^{N-w} \left( \frac{N}{w} \right)^w.$$

Moreover

$$(1 - \theta)^{N-w\theta} \leq \left( \frac{N - w}{N} \right)^{N-w} \left( \frac{w}{N} \right)^w.$$

Combining this yields for the individual redundancy for $N - w \to \infty$ and $w \to \infty$ that

$$\rho(x_1^N) = L(x_1^N) - \log_2 \frac{1}{(1 - \theta)^{N-w\theta}}$$

$$\approx \log_2 \sqrt{\frac{N}{2\pi}} - \log_2 \sqrt{\left( \frac{N - w}{N} \right) \left( \frac{w}{N} \right)} + 2 \text{ bits}.$$
For the expected redundancy, when $0 < \theta < 1$, for large $N$ we obtain

$$\rho = E[L(X_1^N)] - Nh(\theta) \leq \log_2 \sqrt{\frac{N}{2\pi}} - \log_2 \sqrt{(1-\theta)\theta} + 2 \text{ bits}. $$

**UNIVERSAL!**

NOT very good at the borders ...
How to Code the Weight? Krichevsky-Trofimov

**Definition (KT Probability (1981))**

For all integer $a \geq 0$ and $b \geq 0$, define the KT-probability as

$$P_{kt}(a, b) \triangleq \frac{(2a)! \cdot (2b)!}{2^a \cdot a! \cdot 2^b \cdot b!} \cdot \frac{1}{2^{a+b} \cdot (a+b)!}.$$ 

Since $\sum_{w=0}^{N} \binom{N}{w} P_{kt}(N - w, w) = 1$, the weight $w$ of sequence $x_1^N$ can be encoded with a variable-length KT-code of length

$$L_{KT}(w) = \left\lceil \log_2 \frac{1}{\binom{N}{w} P_{kt}(N - w, w)} \right\rceil.$$ 

Now we can write for the total codeword length for sequence $x_1^N$ having weight $w$

$$L(x_1^N) = L_{KT}(w) + L_I(x_1^N)$$

$$= \left\lceil \log_2 \frac{1}{\binom{N}{w} P_{kt}(N - w, w)} \right\rceil + \left\lceil \log_2 \binom{N}{w} \right\rceil$$

$$\leq \log_2 \frac{1}{P_{kt}(N - w, w)} + 2.$$
Assume that our sequence $x_1^N$ has weight $w = w(x_1^N)$, then for all $0 \leq \theta \leq 1$, we get

$$\rho(x_1^N) = L(x_1^N) - \log_2 \frac{1}{(1 - \theta)^{N-w} \theta^w} \leq \log_2 \frac{1}{P_{kt}(N-w, w)} + 2 - 2 \log_2 \frac{1}{(1 - \theta)^{N-w} \theta^w}$$

If both $N - w \to \infty$ and $w \to \infty$ the Stirling approximation yields

$$\frac{1}{P_{kt}(N - w, w)} \approx \sqrt{\frac{\pi N}{2}} \left(\frac{N}{N - w}\right)^{N-w} \left(\frac{N}{w}\right)^w.$$ 

Again

$$(1 - \theta)^{N-w} \theta^w \leq \left(\frac{N - w}{N}\right)^{N-w} \left(\frac{w}{N}\right)^w.$$ 

Combining this yields for the individual redundancy for $N - w \to \infty$ and $w \to \infty$ that

$$\rho(x_1^N) = L(x_1^N) - \log_2 \frac{1}{(1 - \theta)^{N-w} \theta^w} \leq \frac{1}{2} \log_2 N + \frac{1}{2} \log_2 \frac{\pi}{2} + 2 \text{ bits.}$$

KT Coding: Individual Redundancy
KT Coding: Expected Redundancy

For the expected redundancy, when $0 < \theta < 1$, for large $N$ we get

$$\rho = E[L(X_1^N)] - Nh(\theta) \leq \frac{1}{2} \log_2 N + \frac{1}{2} \log_2 \frac{\pi}{2} + 2 \text{ bits.}$$

**NOTE**

It can be shown for KT coding, for all $0 \leq \theta \leq 1$, that

$$\rho(x_1^N) \leq \frac{1}{2} \log_2 N + 3 \text{ bits for all } x_1^N, \text{ and}$$

$$\rho \leq \frac{1}{2} \log_2 N + 3 \text{ bits.}$$
Codewordlengths are roughly $\log_2 N$. Note that $KT$-codewordlength is smaller ($\approx \frac{1}{2} \log_2 N$) for $w = 0$ and $w = N$. 
Individual redundancies are roughly $\frac{1}{2} \log_2 N$. Note that uniform redundancy is larger ($= \log_2 (N + 1)$) for $\theta = 0$ and $\theta = 1$. 
Expected redundancies are roughly $\frac{1}{2} \log_2 N$. Again the uniform redundancy is larger ($= \log_2(N+1)$) for $w = 0$ and $w = N$. 

Example ($N = 255$, expected redundancy as function of $\theta$.)
Conclusion Enumerative Source Coding

- **NON-UNIVERSAL:** $\theta$-Coding yields individual redundancy not exceeding 2 bits.
- **UNIVERSAL:**
  - Both Uniform Coding and KT Coding achieve individual redundancy roughly $\frac{1}{2} \log_2 N$ if both $N - w \to \infty$ and $w \to \infty$.
  - KT Coding has similar behaviour at borders ($w = 0$ and $w = N$).
  - Uniform Coding is simpler.
Arihmetic Coding: Idea Elias

Codewords can be COMPUTED SEQUENTIALLY from the source sequence using conditional PROBABILITIES of next symbol given the previous ones, and vice versa.

\[ P(x_n|x_1 \cdots x_{n-1}) \quad \text{for } n = 1, N \]

\[ P(x_n|x_1 \cdots x_{n-1}) \quad \text{for } n = 1, N \]
If the actual probabilities $P(x_1^N)$ are not known arithmetic coding is still possible if instead of $P(x_1^N)$ we use coding probabilities $P_c(x_1^N)$ satisfying

$$P_c(x_1^N) \geq 0 \text{ for all } x_1^N, \text{ and}$$
$$\sum_{x_1^N} P_c(x_1^N) = 1.$$

Then

$$L(x_1^N) < \log_2 \frac{1}{P_c(x_1^N)} + 2.$$

**PROBLEM:** How do we choose the coding probabilities $P_c(x_1^N)$?
Arithmetic Coding for a Binary IID Source, Unknown $\theta$

Arithmetic Coding based on KT Probability

A good coding probability $P_c(x_1^n)$ for a sequence $x_1^n$ that contains $a$ zeroes and $b = N - a$ ones is

$$P_{kt}(a, b) \triangleq \frac{(2a)!}{2^a a!} \frac{(2b)!}{2^b b!} \frac{1}{2^{a+b}(a+b)!}.$$

Probability of a sequence with $a$ zeroes and $b$ ones followed by a one

$$P_{kt}(a, b + 1) = \frac{b + 1/2}{a + b + 1} \cdot P_{kt}(a, b),$$

hence SEQUENTIAL COMPUTATION is possible!

Using arithmetic coding, the total individual redundancy

$$\rho(x_1^n) < \log_2 \frac{\theta^a (1 - \theta)^b}{P_{kt}(a, b)} + 2 \leq \frac{1}{2} \log_2 N + 3 \text{ bits.}$$

for all $\theta$ and $x_1^n$ with $a$ zeroes and $b$ ones.
OBJECTIVE:
Design good code probabilities for sources with UNKNOWN PARAMETERS and STRUCTURE.
Context Tree Weighting: Binary Tree-Sources

Definition

\( \theta_{10} = 0.3 \)

\( \theta_{00} = 0.5 \)

\( \theta_1 = 0.1 \)

\( \lambda \)

(parameters)

\( (\text{tree-}) \text{ model } M = \{00, 10, 1\} \)

\[ P(X_n = 1 | \cdots, X_{n-1} = 1) = 0.1 \]
\[ P(X_n = 1 | \cdots, X_{n-2} = 1, X_{n-1} = 0) = 0.3 \]
\[ P(X_n = 1 | \cdots, X_{n-2} = 0, X_{n-1} = 0) = 0.5 \]
Node $s$ contains the sequence of source symbols that have occurred following context $s$. Depth is $D$. 

Definition (Context Tree)
Context-Tree Splits Up Sequences in Subsequences

\[
\begin{array}{cccccccccccccc}
1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & \ldots
\end{array}
\]

past

\[
X_1^N
\]
Context-Tree Nodes Containing IID Sequences

Tree-model $\mathcal{M} = \{00, 10, 1\}$
Recursive Definition: From Leaves via Internal Nodes to Root

Let $a_s$ and $b_s$ be the number of zeros resp. ones in the subsequence corresponding to leaf, node, or root $s$.

The subsequence corresponding to a leaf $s$ of the context tree is IID. A good coding probability for this subsequence is therefore

$$P_w(s) \triangleq P_{kt}(a_s, b_s).$$

Weighting the coding probabilities corresponding to both alternatives (node is iid or needs further splitting) yields the coding probability

$$P_w(s) \triangleq \frac{P_{kt}(a_s, b_s) + P_w(0s) \cdot P_w(1s)}{2}$$

for the subsequence that corresponds to node or root $s$. Recursively we find in the root $\lambda$ of the context-tree the coding probability $P_w(\lambda)$ for the entire source sequence $x_1^N$. 
Theorem (W., Shtarkov, and Tjalkens (1995))

*In general for a tree source with \(|\mathcal{M}|\) leaves (parameters):*

\[
\rho(x_1^N) < (2|\mathcal{M}|-1) + \sum_{l \in \mathcal{M}} \frac{1}{2} \log_2(a_l + b_l) + 2 \text{ bits}.
\]

*(model, parameter, and coding redundancies)*

About Model Redundancies

A binary tree model can be described recursively:

- Code(tree) = 0 if tree is only root,
- else Code(tree) = (1, Code(subtree-0), Code(subtree-1)).

This leads to \(2|\mathcal{M}| - 1\) bits.
CTW implements a “weighting” (Bayes mixture) over all tree-models with depth not exceeding $D$, i.e.,

$$P_w(\lambda) = \sum_{\mathcal{M} \leq D} P(\mathcal{M}) P_{kt}(x_1^N|\mathcal{M}),$$

with $P_{kt}(x_1^N|\mathcal{M}) = \prod_{s \in \mathcal{M}} P_{kt}(a_s, b_s)$ and $P(\mathcal{M}) = 2^{-\left(2|\mathcal{M}| - 1\right)}$.

There is one tree-model of depth 0 (i.e. the IID model). If there are $\#_d$ models of depth not exceeding $d$ then there are $\#_d^2 + 1$ models of depth not exceeding $d + 1$. Therefore $\#_2 = 5$, $\#_3 = 26$, $\#_4 = 677$, $\#_5 = 458330$, $\#_6 = 210066388901$, $\#_7 = 4.4128 \cdot 10^{22}$, $\#_8 = 1.9473 \cdot 10^{45}$, etc.

Straightforward analysis. No model-estimation that only gives asymptotic results as in e.g. Rissanen [1983, 1986], Weinberger, Rissanen, and Feder [1995]).

Number of computations needed to process the source sequence $x_1^N$ is linear in $N$. Same holds for the storage complexity.

Optimal parameter redundancy behavior in Rissanen [1984] sense (i.e., $\frac{1}{2} \log_2 N$ bits/parameter).
A modified version achieves entropy not only for tree sources but for all stationary ergodic sources.

A two-pass version (context-tree maximizing) exists that finds the minimum description length (MDL) model, matching to the source sequence. Now

$$P_{m}(s) \triangleq \frac{\max[P_{kt}(a_{s}, b_{s}), P_{m}(0s) \cdot P_{m}(1s)]}{2}.$$  

If a tree source with model generates the sequence $x_{1}^{N}$ the maximizing method produces a model estimate which is correct with probability one as $N \to \infty$. The two-pass version achieves again

$$\rho(x_{1}^{N}) < (2|\mathcal{M}| - 1) + \sum_{l \in \mathcal{M}} \frac{1}{2} \log_{2}(a_{l} + b_{l}) + 2 \text{ bits}.$$  

If instead of $P_{kt}$ we would use “uniform weight coding”

$$P_{u}(x_{1}^{N}) = \frac{1}{N + 1} \left( \frac{N}{w} \right)$$

as coding probability we get similar redundancy results with the problems mentioned before at the borders.
The Burrows Wheeler Transform is a TRANSFORM! The transformed sequence will be coded.

Let $x_1^N = 100101000110$ again. Consider $x_1^N$ and all its cyclic left rotations.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Sort lexicographically based on the suffixes.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

1 1 0 0 1 0 1 0 0 0 1 1 0

3 0 1 0 1 0 0 0 1 1 0 1 0
8 0 0 1 1 0 1 0 0 1 0 1 0
6 1 0 0 0 1 1 0 1 0 0 1 0
4 1 0 1 0 0 0 1 1 0 1 0 0
9 0 1 1 0 1 0 0 1 0 1 0 0
10 1 1 0 1 0 0 1 0 1 0 0 0
Now the left-most column is the BW-transform of $x_1^N$. Hence $BW(100101000110) = (101100110000)$.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

The index $i_{bw}(x_1^N) = 6$ of $x_1^N$ in the ordering is later needed for reconstruction.
We see 7 times 0, and 5 times 1 in the BW-transform, sorting yields the rightmost column.
Observe that, in all rows, a symbol in the leftmost column (BW-transform) follows the symbol in the rightmost column. We then see 3 times 00, 4 times 10, 4 times 01, and 1 times 11. Sorting yields the two rightmost columns.
Again, in all rows, a symbol in the leftmost column (BW-transform) follows the corresponding symbols in the two rightmost columns. We then see 1 times 000, 2 times 100, 3 times 010, 1 times 110, 2 times 001, 2 times 101, 1 times 011, and 0 times 000. Sorting yields the three rightmost columns.
<table>
<thead>
<tr>
<th></th>
<th>. . . . . . . . . .</th>
<th>0</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>. . . . . . . . . .</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>. . . . . . . . . .</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>. . . . . . . . . .</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>. . . . . . . . . .</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>. . . . . . . . . .</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>. . . . . . . . . .</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>. . . . . . . . . .</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>. . . . . . . . . .</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>. . . . . . . . . .</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>. . . . . . . . . .</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
## Burrows Wheeler Reconstruction (5)

<table>
<thead>
<tr>
<th></th>
<th>. . . . . . . .</th>
<th>0 0 1 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>. . . . . . . .</td>
<td>1 1 0 1</td>
</tr>
<tr>
<td>0</td>
<td>. . . . . . . .</td>
<td>0 1 0 1</td>
</tr>
<tr>
<td>0</td>
<td>. . . . . . . .</td>
<td>1 0 0 1</td>
</tr>
<tr>
<td>1</td>
<td>. . . . . . . .</td>
<td>0 0 0 1</td>
</tr>
<tr>
<td>1</td>
<td>. . . . . . . .</td>
<td>0 1 1 0</td>
</tr>
<tr>
<td>0</td>
<td>. . . . . . . .</td>
<td>1 0 1 0</td>
</tr>
<tr>
<td>0</td>
<td>. . . . . . . .</td>
<td>1 0 1 0</td>
</tr>
<tr>
<td>1</td>
<td>. . . . . . . .</td>
<td>0 0 1 0</td>
</tr>
<tr>
<td>1</td>
<td>. . . . . . . .</td>
<td>0 1 0 0</td>
</tr>
<tr>
<td>0</td>
<td>. . . . . . . .</td>
<td>0 1 0 0</td>
</tr>
<tr>
<td>1</td>
<td>. . . . . . . .</td>
<td>1 0 0 0</td>
</tr>
</tbody>
</table>
### Burrows Wheeler Reconstruction (6)

<table>
<thead>
<tr>
<th>0</th>
<th>. . . . . .</th>
<th>0 0 0 1 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>. . . . . .</td>
<td>0 1 1 0 1</td>
</tr>
<tr>
<td>0</td>
<td>. . . . . .</td>
<td>0 0 1 0 1</td>
</tr>
<tr>
<td>0</td>
<td>. . . . . .</td>
<td>0 1 0 0 1</td>
</tr>
<tr>
<td>1</td>
<td>. . . . . .</td>
<td>1 0 0 0 1</td>
</tr>
<tr>
<td>1</td>
<td>. . . . . .</td>
<td>0 0 1 1 0</td>
</tr>
<tr>
<td>0</td>
<td>. . . . . .</td>
<td>1 1 0 1 0</td>
</tr>
<tr>
<td>0</td>
<td>. . . . . .</td>
<td>0 1 0 1 0</td>
</tr>
<tr>
<td>1</td>
<td>. . . . . .</td>
<td>1 0 0 1 0</td>
</tr>
<tr>
<td>1</td>
<td>. . . . . .</td>
<td>1 0 1 0 0</td>
</tr>
<tr>
<td>0</td>
<td>. . . . . .</td>
<td>1 0 1 0 0</td>
</tr>
<tr>
<td>1</td>
<td>. . . . . .</td>
<td>0 1 0 0 0</td>
</tr>
</tbody>
</table>
### Burrows Wheeler Reconstruction (7)

| 0 | . . . . . | 1 0 0 0 1 1 |
| 0 | . . . . . | 0 0 1 1 0 1 |
| 0 | . . . . . | 1 0 0 1 0 1 |
| 0 | . . . . . | 1 0 1 0 0 1 |
| 1 | . . . . . | 0 1 0 0 0 1 |
| 1 | . . . . . | 0 0 0 1 1 0 |
| 0 | . . . . . | 0 1 1 0 1 0 |
| 0 | . . . . . | 0 0 1 0 1 0 |
| 1 | . . . . . | 0 1 0 0 1 0 |
| 1 | . . . . . | 1 1 0 1 0 0 |
| 0 | . . . . . | 0 1 0 1 0 0 |
| 1 | . . . . . | 1 0 1 0 0 0 |
| 0 | . . . . . 0 1 0 0 0 0 1 1 |
| 0 | . . . . . 0 0 0 1 1 0 0 1 |
| 0 | . . . . . 0 1 0 0 1 0 0 1 |
| 0 | . . . . . 1 1 0 1 0 0 0 1 |
| 1 | . . . . . 1 0 1 0 0 0 0 1 |
| 1 | . . . . . 1 0 0 0 1 1 0 0 |
| 0 | . . . . . 0 0 1 1 0 1 0 0 |
| 0 | . . . . . 1 0 0 1 0 1 0 0 |
| 1 | . . . . . 1 0 1 0 0 0 1 0 |
| 1 | . . . . . 0 1 1 0 1 0 0 0 |
| 0 | . . . . . 0 0 1 0 1 0 0 0 |
| 1 | . . . . . 0 1 0 1 0 0 0 0 |
### Burrows Wheeler Reconstruction (9)

<table>
<thead>
<tr>
<th>(0)</th>
<th>(\ldots)</th>
<th>(1)</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0)</td>
<td>(\ldots)</td>
<td>(1)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>(0)</td>
<td>(\ldots)</td>
<td>(1)</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>(0)</td>
<td>(\ldots)</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>(\ldots)</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>(\ldots)</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>(\ldots)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>(\ldots)</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>(\ldots)</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>(\ldots)</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>(\ldots)</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>(\ldots)</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>.</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>0</td>
<td>.</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>.</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>.</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>.</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>.</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>.</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>.</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>.</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>.</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>.</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>.</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Burrows Wheeler Reconstruction (11)

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0 1 0 0 1 0 1 0 0 0 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>-------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0 0 1 0 1 0 0 0 1 1 0 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0 0 0 1 1 0 1 0 0 1 0 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0 1 0 0 0 1 1 0 1 0 0 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1 0 1 0 0 1 0 1 0 0 0 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1 0 0 1 0 1 0 0 0 1 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0 1 0 1 0 0 0 1 1 0 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0 0 1 1 0 1 0 0 1 0 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1 0 0 0 1 1 0 1 0 0 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1 0 1 0 0 0 1 1 0 1 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0 1 1 0 1 0 0 1 0 1 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1 1 0 1 0 0 1 0 1 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

From the index $i_{bw}(x_1^N) = 6$ of $x_1^N$ we get the original sequence back.
Coding the BW-transformed sequence

IDEA:

For tree sources, the transformed sequence $BW(x_1^N)$ is a concatenation of IID subsequences, as many as there are leaves in the tree model.\(^a\)

\(^a\)Circularity effects are ignored.

Example ($x_1^N = 100101000110$ and tree-model $\mathcal{M} = \{00, 10, 1\}$)

$BW(100101000110) = 101, 1001, 10000$. See

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
IDEA:

For tree sources, the transformed sequence $BW(x_1^N)$ is a concatenation of IID subsequences, as many as there are leaves in the tree model.\(^a\)

\(^a\)Circularity effects are ignored.

**Example ($x_1^N = 100101000110$ and tree-model $M = \{00, 10, 1\}$)**

$BW(100101000110) = 101, 1001, 10000$. See

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Move-to-front techniques (Bentley et al. (1986), Elias (1987)) combined with Huffman codes can be used.

Piecewise IID coding (Merhav (1993), W. (1996)) can be used. These methods demonstrate that we need to specify the transition points, which requires roughly $\log_2 N$ bits per transition (Effros et al. (2002)).

CTW only needs $2|\mathcal{M}| - 1$ bits, which is roughly 2 bits per transition.

**OBJECTIVE:**

Design a procedure for coding the BW transformed sequence that needs only $2|\mathcal{M}| - 1$ bits.
FSM-Closed Tree Models

**Definition**

The generator of leaf $u_d u_{d-1} \cdots u_1$ at depth $d$ is $u_d u_{d-1} \cdots u_2$ at depth $d-1$. A tree model is **FSM-closed** if all its leaves have a generator that is either a leaf or internal node of the tree model.

**Example (Tree model $\mathcal{M} = \{00, 010, 110, 1\}$)**

Leaf 00 has generator 0 which is an internal node of $\mathcal{M}$. But note that leaves 110 and 010 do not have a generator in tree model $\mathcal{M}$. Therefore $\mathcal{M}$ is not FSM-closed and we cannot describe the source as a FSM.
Example (Tree model $\mathcal{M}' = \{00, 010, 110, 01, 11\}$)

Tree model $\mathcal{M}'$ is FSM-closed.

If a tree model is FSM-closed then each leaf and each node in the tree model has a generator that is a leaf or node of the tree model.
FSM-Closed Tree Models

Definition
A tree model can be made FSM-closed by adding the generators of all leaves. The resulting model is the FSM-closure of the tree model.

Example
![Example Diagram](image_url)
Definition
A tree model can be made FSM-closed by adding the generators of all leaves. The resulting model is the FSM-closure of the tree model.

Example

```
   11
  /   \
 /     \
01     1
  \
   \
00 010 110
   \
    \
   0 10

\lambda
```
Coding BW-sequences: PROBLEM

Suppose that \( \hat{M} \) is the tree-model that matches best to the BW-sequence. More about how to find \( \hat{M} \) later.

- This tree model \( \hat{M} \) is included in the description \((2|\hat{M}| - 1 \text{ bits})\).
- Apart from that the \( b \)-counts (weights of subsequences) of all leaves of \( \hat{M} \) are added to the description.
- Finally the lexicographical indices of all the subsequences corresponding to the leaves of \( \hat{M} \) are included.

**Question:**
Can we reconstruct the entire BW-sequence from the description using ENUMERATIVE techniques?

**Definition**
Let \( a_s \) the number of zeros and \( b_s \) the number of ones in the subsequence that corresponds to \( s \), then

\[
    c_s = a_s + b_s
\]
is length of this subsequence.
Suppose that the $b$-counts in the leaves of **FSM-closed tree model** $\hat{M} = \{00, 010, 110, 01, 11\}$ are given to the decoder, hence the decoder knows $b_{00}$, $b_{010}$, $b_{110}$, $b_{01}$, $b_{11}$, and $N$.

- The decoder first computes the $b$-counts in all the nodes of the tree model, hence $b_\lambda$, $b_0$, $b_1$, and $b_{01}$.
- The decoder now processes layer by layer, starting in the root (layer 0). First in the root the $a$-count is computed.

\[
a_\lambda = N - b_\lambda.
\]

Layer 0 is processed now.

- In layer 1 there are two nodes, 0 and 1 that fit into $\hat{M}$. Since $\hat{M}$ is FSM-closed, their generator $\lambda$ exists, is at level 0, and is thus processed before. Therefore we can compute the $c$'s of these nodes.

\[
c_0 = a_\lambda \\
c_1 = b_\lambda.
\]

With the available $b$-counts also the $a$-counts can be computed

\[
a_0 = c_0 - b_0 \\
a_1 = c_1 - b_1.
\]

Layer 1 is processed now.
 FSM-Closed Tree Model (2)
FSM-Closed Tree Model (3)
FSM-Closed Tree Model (4)
FSM-Closed Tree Model (5)

\[ \begin{align*}
N & \quad a_\lambda \\
& \quad b_\lambda \\
& \quad c_0 \\
& \quad b_0 \\
& \quad b_{01} \\
& \quad b_{10} \\
& \quad b_{11} \\
\end{align*} \]
FSM-Closed Tree Model (6)
In layer 2 there are three leaves, 00, 01, and 11, and a node 10, that fit into $\hat{M}$. Since $\hat{M}$ is FSM-closed, again their generators 0 and 1 exist, are at level 1, and are thus processed before. Now we can compute the $c$'s of these leaves and node.

\[
\begin{align*}
    c_{00} &= a_0 \\
    c_{10} &= a_1 \\
    c_{01} &= b_0 \\
    c_{11} &= b_1.
\end{align*}
\]

With the available $b$-counts also the $a$-counts can be computed for these leaves and node

\[
\begin{align*}
    a_{00} &= c_{00} - b_{00} \\
    a_{10} &= c_{10} - b_{10} \\
    a_{01} &= c_{01} - b_{01} \\
    a_{11} &= c_{11} - b_{11}.
\end{align*}
\]
FSM-Closed Tree Model (8)
FSM-Closed Tree Model (9)

\[
\begin{align*}
&c_{11} \quad a_{11} \quad b_{11} \\
&c_{01} \quad a_{01} \quad b_{01} \\
&c_{10} \quad a_{10} \quad b_{10} \\
&c_{00} \quad a_{00} \quad b_{00} \\
&N \quad a_\lambda \quad b_\lambda \\
&b_{110} \\
&b_{010} \\
\end{align*}
\]
In layer 3 there are two leaves, 010, and 110, that fit into \( \hat{M} \). Since \( \hat{M} \) is FSM-closed, again their generators 01 and 11 exist, are at level 2, and are thus processed before. Now we can compute the \( c \)'s of the subsequences in these two leaves.

\[
\begin{align*}
c_{010} &= a_{01} \\
c_{110} &= a_{11}.
\end{align*}
\]

With the available \( b \)-counts also the \( a \)-counts can be computed for these leaves

\[
\begin{align*}
a_{010} &= c_{010} - b_{010} \\
a_{110} &= c_{110} - b_{110}.
\end{align*}
\]
FSM-Closed Tree Model (11)
FSM-Closed Tree Model (12)
For all leaves $s$ in $\hat{M}$ the subsequence length $c_s$ and $b$-count $b_s$ is known. Note that the $b$-count is the weight of the subsequence.

Now with the indices $i_{00}$, $i_{010}$, $i_{110}$, $i_{01}$, and $i_{11}$, the corresponding subsequences $bw_{00}$, $bw_{010}$, $bw_{110}$, $bw_{01}$, and $bw_{11}$, can be reconstructed.

The BW-transformed sequence is now the concatenation of the five subsequences, hence

$$BW = bw_{00}, bw_{010}, bw_{110}, bw_{01}, bw_{11}.$$  

NOTE that this approach is outlined in Martin, Seroussi, & Weinberger (2004), with emphasis on the FSM-case, not on the BWT-case.
Suppose that the $b$-counts in the leaves of tree model $\hat{M} = \{00, 010, 110, 1\}$, which is not FSM-closed, are given to the decoder, hence the decoder knows $b_{00}, b_{010}, b_{110}, b_1,$ and $N$.

- The decoder first computes the $b$-counts in all the nodes of the tree model, hence $b_\lambda, b_0,$ and $b_{10}$.
- The decoder now processes layer by layer, starting in the root (layer 0). First in the root the $a$-count is computed.
  \[ a_\lambda = N - b_\lambda. \]

Now all the nodes in layer 0 are processed.

- In layer 1 there are two nodes, 0 and 1, that fit into $\hat{M}$. Since “all nodes in layer 0” are complete, we can compute the $c$’s of the nodes 0 and 1.
  \[ c_0 = a_\lambda \]
  \[ c_1 = b_\lambda. \]

With the available $b$-counts also the $a$-counts can be computed for these nodes
  \[ a_0 = c_0 - b_0 \]
  \[ a_1 = c_1 - b_1. \]

Now all the nodes in layer 1 are processed.
Tree Model not FSM-Closed (2)
Tree Model not FSM-Closed (3)
Tree Model, not FSM-Closed (4)
Tree Model, not FSM-Closed (6)
In layer 2 there is a leaf, 00, and a node 10, that fit into \( \hat{M} \). Since all nodes at layer 1 are complete, we can compute the \( c \)'s of the subsequences in this leaf and node.

\[
\begin{align*}
c_{00} &= a_0 \\
c_{10} &= a_1.
\end{align*}
\]

With the available \( b \)-counts for the node and leaf that fit into \( \hat{M} \), the \( a \)-counts can be computed

\[
\begin{align*}
a_{00} &= c_{00} - b_{00} \\
a_{10} &= c_{10} - b_{10}.
\end{align*}
\]

In layer 2 there are two nodes, 01 and 11, that do not fit into \( M \). Since all nodes at layer 1 are processed, we can also compute the \( c \)'s of these two nodes.

\[
\begin{align*}
c_{01} &= b_0 \\
c_{11} &= b_1.
\end{align*}
\]

We need \( b_{01} \) and \( b_{11} \) now ...
New Approach

- Leaf 1 at layer 1 is complete (we know \( c_1 \) and weight \( b_1 \)), and therefore the corresponding subsequence \( bw_1 \) can be reconstructed from the lexicographical index \( i_1 \).

- The first \( c_{01} \) digits of \( bw_1 \) are digits corresponding to node 01, call this sequence \( bw_{01} \). We can now simply count the number \( a_{01} \) of 0-digits and the number \( b_{01} \) of 1-digits in \( bw_{01} \).

- The last \( c_{11} \) digits of \( bw_1 \) are digits corresponding to node 11, call this sequence \( bw_{11} \). We can now simply count the number \( a_{11} \) of 0-digits and the number \( b_{11} \) of 1-digits in \( bw_{11} \).

Now all the nodes in layer 2 are processed.
Tree Model, not FSM-Closed (9)
Tree Model, not FSM-Closed (10)
Tree Model, not FSM-Closed (11)
In layer 3 there are two leaves, 010, and 110, that fit into \( \mathcal{M} \). Since all nodes at level 2 are complete, we can compute the \( c \)'s of these leaves.

\[
c_{010} = a_{01} \\
c_{110} = a_{11}.
\]

With the available \( b \)-counts also the \( a \)-counts can be computed for these leaves and node

\[
a_{010} = c_{010} - b_{010} \\
a_{110} = c_{110} - b_{110}.
\]
Tree Model, not FSM-Closed (13)
Tree Model, not FSM-Closed (14)
For all remaining leaves $s$ in $\widehat{M}$ the subsequence length $c_s$ and $b$-count $b_s$ (weight) is known.

Now with the indices $i_{00}$, $i_{010}$, and $i_{110}$, the corresponding subsequences $bw_{00}$, $bw_{010}$, and $bw_{110}$, can be reconstructed.

Since $bw_1$ was reconstructed before, the BW-transformed sequence is now the concatenation of the four subsequences, hence

$$BW = bw_{00}, bw_{010}, bw_{110}, bw_1.$$
Coding $b$-Counts: Objective

How do we code the $b$-counts in the leaves $L(\widehat{M})$? Ideally we would like to achieve total codewordlength

$$\sum_{l \in L(\widehat{M})} \log_2(a_l + b_l + 1).$$

This corresponds to uniform weight coding. Note that the $c$’s apart from $c_\lambda = N$, are still unknown.
Coding $b$-Counts: Procedure

- We start in the root node $\lambda$. There we describe $b_{\lambda}$. To this we need
  $$\left\lceil \log_2(a_{\lambda} + b_{\lambda} + 1) \right\rceil$$ bits.

- Then in all internal nodes $s$, starting in the root node $\lambda$, and processing level by level, we first find
  $$i = \arg \min_{i=0,1} \log_2(a_{is} + b_{is} + 1),$$
  and we use one bit to describe $i$. Then we describe $b_{is}$ using
  $$\left\lceil \log_2(a_{is} + b_{is} + 1) \right\rceil$$ bits.

Note that when $b_{is}$ needs to be described, the corresponding $c_{0s} = a_{0s} + b_{0s}$ and $c_{1s} = a_{1s} + b_{1s}$ have already been reconstructed.

This procedure leads to
$$\left\lceil \log_2(a_{\lambda} + b_{\lambda} + 1) \right\rceil + \sum_{n \in N(\hat{\mathcal{M}})} \left\lceil \log_2 \min(a_{0n} + b_{0n} + 1, a_{1n} + b_{1n} + 1) \right\rceil$$ bits,

where $N(\hat{\mathcal{M}})$ are the internal nodes of $\hat{\mathcal{M}}$. 
Consider a node \( n \in \mathcal{N} (\widehat{\mathcal{M}}) \).

- Observe that we have described \( b_n \) using \( \lceil \log_2 (c_n + 1) \rceil \) bits, and next we describe both \( b_{0n} \) and \( b_{1n} \) using
  \[ 1 + \lceil \log_2 \min(c_{0n} + 1, c_{1n} + 1) \rceil \]
  bits.
- Directly describing \( b_{0n} \) and \( b_{1n} \) would require
  \( \lceil \log_2 (c_{0n} + 1) \rceil + \lceil \log_2 (c_{1n} + 1) \rceil \) bits.
- The loss in this node is now
  \[
  \lceil \log_2 (c_n + 1) \rceil + 1 + \lceil \log_2 \min(c_{0n} + 1, c_{1n} + 1) \rceil - \lceil \log_2 (c_{0n} + 1) \rceil - \lceil \log_2 (c_{1n} + 1) \rceil \\
  = \lceil \log_2 (c_n + 1) \rceil + 1 - \lceil \log_2 \max(c_{0n} + 1, c_{1n} + 1) \rceil \\
  \leq 2 \text{ bit}.
  
  Total loss is therefore
  \[ 2(|\widehat{\mathcal{M}}| - 1) \text{ bits.} \]
Finding Best Tree Model: Maximizing

A two-pass version (context-tree maximizing) exists that finds the best model (MDL) matching to the source sequence, if coding is done as described before.

Context-Tree Maximizing

Define for nodes $n$

$$
\mu(n) \triangleq 1 + \min\left[ \log_2 \left( \frac{c_n}{b_n} \right) \right], 1 + \left\lceil \log_2 \min(c_{0n} + 1, c_{1n} + 1) \right\rceil + \mu(0n) + \mu(1n),
$$

while for leaves $l$

$$
\mu(l) \triangleq \left\lceil \log_2 \left( \frac{c_l}{b_l} \right) \right\rceil.
$$

Tracking this procedure yield $\hat{M}$. Code-length is

$$
L = \log_2(N + 1) + \mu(\lambda).
$$

This procedure can be carried out very efficiently during the BW transform phase.
Finding Best Tree Model: Integration with BW Transform

Fix a depth, e.g. 3, then the subsequences of the nodes at depth 3 can be easily found, see BW table below:

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Now node 000 → {10} → 1, node 100 → {9, 4} → 01, node 010 → {6, 8, 3} → 100, node 110 → {1} → 1, node 001 → {5, 11} → 10, node 101 → {7, 2} → 00, node 011 → {12} → 0, and finally node 111 → ∅ → ∅.
Binary Decomposition

CTW: Binary Decomposition, 1.8 bit/ASCII.

- Bytes: 8 bits,
  - ASCII-symbols: 7 bits,
  - DNA nucleotide (A, T, G and C): 2 bits, etc.

- The first bit of a symbol depends on the context (a number of past symbols).
  - The second bit of a symbol depends on the first bit of that symbol and the context.
  - The third bit of a symbol depends on the first and second bit of that symbol and the context, etc.

- Therefore there is a tree model for the first bit.
  - There are two tree models for the second bit, one for first bit being 0 and a second one when the first bit is 1.
  - There are four tree models for the third bit etc.

Question:
Can we do a binary decomposition also in combination with a BW transform on the symbols? Does there exist a way of finding the c’s based on only the b-counts?
**Binary Decomposition: Example (1)**

**SYMBOL-SIZE: 2 bits.**
Suppose that the $b$-counts in the leaves of the three tree models are specified. The decoder can now compute the $b$-counts of all inner nodes.

- The decoder first computes in tree-$\emptyset$ (corresponding to the first digit of the symbol) count $a_\lambda^\emptyset$

$$a_\lambda^\emptyset = N - b_\lambda^\emptyset.$$ 

Now in tree-$0$ and in tree-$1$ (corresponding to the second digit of the symbol) the decoder can compute the $c$'s and then the $a$'s in the root nodes

$$c_\lambda^0 = a_\lambda^\emptyset \quad a_\lambda^0 = c_\lambda^0 - b_\lambda^0 \quad c_\lambda^1 = b_\lambda^\emptyset \quad a_\lambda^1 = c_\lambda^1 - b_\lambda^1.$$ 

Now all the nodes in layer 0 of the three trees are processed.

- The decoder now starts working on layer 1 (one symbol contexts).
  First in tree-$\emptyset$ the decoder computes the $c$'s and then the $a$'s

$$c_{00}^\emptyset = a_\lambda^\emptyset \quad a_{00}^\emptyset = c_{00}^\emptyset - b_{00}^\emptyset$$
$$c_{01}^\emptyset = b_\lambda^\emptyset \quad a_{01}^\emptyset = c_{01}^\emptyset - b_{01}^\emptyset$$
$$c_{10}^1 = a_\lambda^1 \quad a_{10}^1 = c_{10}^1 - b_{10}^1$$
$$c_{11}^1 = b_\lambda^1 \quad a_{11}^1 = c_{11}^1 - b_{11}^1$$
Then in tree-0 the decoder computes the $c$’s and then the $a$’s

\[
\begin{align*}
    c_{00}^0 &= a_{00}^0 & a_{00}^0 &= c_{00}^0 - b_{00}^0 \\
    c_{01}^0 &= a_{01}^0 & a_{01}^0 &= c_{01}^0 - b_{01}^0 \\
    c_{10}^0 &= a_{10}^0 & a_{10}^0 &= c_{10}^0 - b_{10}^0 \\
    c_{11}^0 &= a_{11}^0 & a_{11}^0 &= c_{11}^0 - b_{11}^0,
\end{align*}
\]

and in tree-1 the decoder computes the $c$’s and then the $a$’s

\[
\begin{align*}
    c_{00}^1 &= b_{00}^1 & a_{00}^1 &= c_{00}^1 - b_{00}^1 \\
    c_{01}^1 &= b_{01}^1 & a_{01}^1 &= c_{01}^1 - b_{01}^1 \\
    c_{10}^1 &= b_{10}^1 & a_{10}^1 &= c_{10}^1 - b_{10}^1 \\
    c_{11}^1 &= b_{11}^1 & a_{11}^1 &= c_{11}^1 - b_{11}^1.
\end{align*}
\]

Now all the nodes in layer 1 of the three trees are processed.

Etc.

If a leaf is encountered the corresponding subsequence is reconstructed using the index. From then on $a$-counts and $b$-counts can be found by inspection of the reconstructed subsequence.
Binary Decomposition: Example (3)
Binary Decomposition: Example (4)
Binary Decomposition: Example (6)
Binary Decomposition: Example (7)
Binary Decomposition: Example (8)
INTRODUCTION
IID SOURCES, PREFIX CODES, REDUNDANCY
ENUMERATIVE CODING
ARITHMETIC CODING
CONTEXT-TREE WEIGHTING
BURROWS WHEELER CODING BW-SEQUENCES
CODING B-COUNTS
FIND BEST TREE MODEL
BINARY DECOMPOSITION
Problem
Example
Coding b-Counts, Maximizing
CONCLUSION
FUTURE DIRECTIONS

Binary Decomposition: Example (9)
Binary Decomposition: Example (10)
Binary Decomposition: Coding $b$-Counts, Maximizing

- Coding the $b$-counts costs more bits now. It can be shown that we loose at most 6 bit per internal node (quaternary splitting can be accomplished with three binary splits). Fortunately this is at most 2 bit per parameter again.
- Maximizing formula exists again.
The BW transform method does not need $\log_2(N)$ bits to specify a transition.

The FSM closure is not needed. Specifying the maximizing tree model $\hat{M}$ and the $b$-counts in the leaves is enough.

A loss comes from the fact that these $b$-counts are specified recursively. This loss is upper-bounded by 2 bit per leaf.

Maximizing methods exist that match perfectly with the BW transform.

There exist also binary decomposition techniques that combine with the BW transform.

Also KT-coding of weights can be analysed.
Future Directions

- Software?
- Suppose that the data have left-right symmetry hence
  \[ P(a, b) = P(b, a), \quad P(a, b, c) = P(c, b, a), \]
  \[ P(a, b, c, d) = P(d, c, b, a), \] etc. This reduces the number of parameters. Algorithm? Important for image-compression.
- CTW can handle side-information by considering it as context (e.g. Cai, Kulkarni, and Verdu, 2005). BW-version?
- Can BW-techniques be used to achieve CT-weighting performance. Here BW-techniques are described that achieve CT-maximizing performance.
- What if the side-information is not-properly aligned? LZ is more robust now. Important for reference-based genome compression (Chern et al., 2012).