Lossless Source Coding Algorithms

Frans M.J. Willems

Department of Electrical Engineering
Eindhoven University of Technology

ISIT 2013, Istanbul, Turkey
INTRODUCTION

HUFFMAN and TUNSTALL
- Binary IID Sources
- Huffman Code
- Tunstall Code

ENUMERATIVE CODING
- Lexicographical Ordering
  - FV: Pascal-\(\Delta\) Method
  - VF: Petry Code

ARITHMETIC CODING
- Intervals
  - Universal Coding, Individual Redundancy

CONTEXT-TREE WEIGHTING
- IID, unknown \(\theta\)
- Binary Tree-Sources
- Context Trees
  - Coding Probabilities

REPETITION TIMES
- LZ77
- Repetition Times, Kac
  - Repetition-Time Algorithm
  - Achieving Entropy

CONCLUSION
LOSSLESS SOURCE CODING ALGORITHMS

Frans M.J. Willems

INTRODUCTION

HUFFMAN-TUNSTALL
Binary IID Sources
Huffman Code
Tunstall Code

ENUMERATIVE CODING
Lexicographical Ordering
FV: Pascal-∆ Method
VF: Petry Code

ARITHMETIC CODING
Intervals
Universal Coding, Individual Redundancy

CONTEXT-TREE WEIGHTING
IID, unknown θ
Tree Sources
Context Trees
Coding Prbs., Redundancy

REPETITION TIMES
LZ77
Repetition Times, Kac
Repetition-Time Algorithm
Achieving Entropy

CONCLUSION

POSSIBLE TOPICS:

- Multi-user Information Theory (with Edward van der Meulen (KUL), Andries Hekstra)
- Lossless Source Coding (with Tjalling Tjalkens, Yuri Shtarkov (IPPI), Paul Volf)
- Watermarking, Embedding, and Semantic Coding (with Martin van Dijk, Ton Kalker (Philips Research))
- Biometrics (with Tanya Ignatenko)

LOSSLESS SOURCE CODING ALGORITHMS

WHY?

- Not many sessions at ISIT 2012! Is lossless source coding DEAD?
- Lossless Source Coding is about UNDERSTANDING data. Universal Lossless Source Coding is focussing on FINDING STRUCTURE in data. MDL principle [Rissanen].
- ALGORITHMS are fun (Piet Schalkwijk).
LOSSLESS SOURCE CODING ALGORITHMS
Frans M.J. Willems

INTRODUCTION
HUFFMAN-TUNSTALL
Binary IID Sources
Huffman Code
Tunstall Code

ENUMERATIVE CODING
Lexicographical Ordering
FV: Pascal-∆ Method
VF: Petry Code

ARITHMETIC CODING
Intervals
Universal Coding, Individual Redundancy

CONTEXT-TREE WEIGHTING
IID, unknown θ
Tree Sources
Context Trees
Coding Prbs., Redundancy

REPETITION TIMES
LZ77
Repetition Times, Kac
Repetition-Time Algorithm
Achieving Entropy

CONCLUSION

POSSIBLE TOPICS:

- Multi-user Information Theory (with Edward van der Meulen (KUL), Andries Hekstra)
- Lossless Source Coding (with Tjalling Tjalkens, Yuri Shtarkov (IPPI), Paul Volf)
- Watermarking, Embedding, and Semantic Coding (with Martin van Dijk, Ton Kalker (Philips Research))
- Biometrics (with Tanya Ignatenko)

LOSSLESS SOURCE CODING ALGORITHMS

WHY?

- Not many sessions at ISIT 2012! Is lossless source coding DEAD?
- Lossless Source Coding is about UNDERSTANDING data. Universal Lossless Source Coding is focussing on FINDING STRUCTURE in data. MDL principle [Rissanen].
- ALGORITHMS are fun (Piet Schalkwijk).
POSSIBLE TOPICS:

- Multi-user Information Theory (with Edward van der Meulen (KUL), Andries Hekstra)
- Lossless Source Coding (with Tjalling Tjalkens, Yuri Shtarkov (IPPI), Paul Volf)
- Watermarking, Embedding, and Semantic Coding (with Martin van Dijk, Ton Kalker (Philips Research))
- Biometrics (with Tanya Ignatenko)

LOSSLESS SOURCE CODING ALGORITHMS

WHY?

- Not many sessions at ISIT 2012! Is lossless source coding DEAD?
- Lossless Source Coding is about UNDERSTANDING data. Universal Lossless Source Coding is focussing on FINDING STRUCTURE in data. MDL principle [Rissanen].
- ALGORITHMS are fun (Piet Schalkwijk).
Lecture Structure

- **Tutorial**, binary case, my favorite algorithms, ...
- **Remarks**, open problems, ...

**Lossless Source Coding Algorithms**
Frans M.J. Willems

**Introduction**
Huffman-Tunstall
Binary IID Sources
Huffman Code
Tunstall Code

**Enumerative Coding**
Lexicographical Ordering
FV: Pascal-Δ Method
VF: Petry Code

**Arithmetic Coding**
Intervals
Universal Coding, Individual Redundancy

**Context-Tree Weighting**
IID, unknown $\theta$
Tree Sources
Context Trees
Coding Prbs., Redundancy

**Repetition Times**
LZ77
Repetition Times, Kac
Repetition-Time Algorithm
Achieving Entropy

**Conclusion**
INTRODUCTION

HUFFMAN and TUNSTALL
- Binary IID Sources
- Huffman Code
- Tunstall Code

ENUMERATIVE CODING
- Lexicographical Ordering
- FV: Pascal-∆ Method
- VF: Petry Code

ARITHMETIC CODING
- Intervals
- Universal Coding, Individual Redundancy

CONTEXT-TREE WEIGHTING
- IID, unknown \( \theta \)
- Binary Tree-Sources
- Context Trees
- Coding Probabilities

REPETITION TIMES
- LZ77
- Repetition Times, Kac
- Repetition-Time Algorithm
- Achieving Entropy

CONCLUSION
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

$$P(1) = \theta, \text{ and } P(0) = 1 - \theta.$$

A sequence $x_1^N$ containing $N - w$ zeros and $w$ ones has probability

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

**Entropy 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).
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

$$P(1) = \theta, \text{ and } P(0) = 1 - \theta.$$

A sequence $x_1^N$ containing $N - w$ zeros and $w$ ones has probability

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

**Entropy 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).
IDEA:
Give more probable sequences shorter codewords than less probable sequences.

Definition (FV-Length Code)

A FV-length code assigns to source sequence \( x_1^N \) a binary codeword \( c(x_1^N) \) of length \( L(x_1^N) \). The rate of a FV code is

\[
R \triangleq \frac{E[L(x_1^N)]}{N} \quad \text{(code-symbols/source-symbol)}.
\]

GOAL:
We would like to find decodable FV-length codes that MINIMIZE this rate.
Definition (Prefix code)

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

We focus on 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>

\[ 
\begin{array}{c|cc}
  \emptyset & 1 & 11 \\
  0 & 10 & 110 \\
  11 & 111 & 111 \\
\end{array} 
\]
### 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

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

or equivalently \( R \geq h(\theta). \) The minimum is achieved if and only if

\[ L(x_1^N) = -\log_2(P(x_1^N)) \]  

(ideal codeword length) for all \( x_1^N \in \mathcal{X}^N \) with nonzero \( P(x_1^N). \)

(b) There exist prefix codes with

\[ E[L(X_1^N)] < H(X_1^N) + 1 = Nh(\theta) + 1, \]

or equivalently \( R < h(\theta) + 1/N. \)
Huffman’s Code

Definition (Optimal FV-length Code)

A code that minimizes the expected codeword-length \( E[L(X_1^N)] \) (and the rate \( R \)) is called **optimal**.

Theorem (Huffman, 1952)

*The Huffman construction leads to an optimal FV-length code.*

**Construction:**

- Consider the set of probabilities \( \{P(x_1^N), x_1^N \in \mathcal{X}^N\} \).
- Replace two smallest probabilities by a probability which is their sum. Label the branches from these two smallest probabilities to their sum with code-symbols “0” and “1”.
- Continue like this until only one probability (equal to 1) is left.

Obviously Huffman’s code results in \( E[L(X_1^N)] < H(X_1^N) + 1 = Nh(\theta) + 1 \) and therefore \( R < h(\theta) + 1/N \).
Huffman’s Code

**Definition (Optimal FV-length Code)**

A code that minimizes the expected codeword-length $E[L(X_1^N)]$ (and the rate $R$) is called **optimal**.

**Theorem (Huffman, 1952)**

*The Huffman construction leads to an optimal FV-length code.*

**CONSTRUCTION:**

- **Consider the set of probabilities** $\{P(x_1^N), x_1^N \in X^N\}$.
- **Replace two smallest probabilities by a probability which is their sum.** Label the branches from these two smallest probabilities to their sum with code-symbols “0” and “1”.
- **Continue like this until only one probability (equal to 1) is left.**

Obviously Huffman’s code results in $E[L(X_1^N)] < H(X_1^N) + 1 = Nh(\theta) + 1$ and therefore $R < h(\theta) + 1/N$. 
Huffman’s Construction

Example

Let $N = 3$ and $\theta = 0.3$, then $h(0.3) = 0.881$.

Now $E[L(X_1^N)] = 4(0.027 + 0.063 + 0.063 + 0.063) + 3(0.147 + 0.147) + 2(0.147 + 0.343) = 2.726$. Therefore $R = 2.726/3 = 0.909$. 
Huffman’s Construction

Example

Let $N = 3$ and $\theta = 0.3$, then $h(0.3) = 0.881$.

Now $E[L(X_1^N)] = 4(0.027 + 0.063 + 0.063 + 0.063) + 3(0.147 + 0.147) + 2(0.147 + 0.343) = 2.726$. Therefore $R = 2.726/3 = 0.909$. 
Huffman’s Construction

Let $N = 3$ and $\theta = 0.3$, then $h(0.3) = 0.881$.

Now $E[L(X_1^N)] = 4(.027 + .063 + .063 + .063) + 3(.147 + .147) + 2(.147 + .343) = 2.726$. Therefore $R = 2.726/3 = 0.909$. 
Huffman’s Construction

Example

Let $N = 3$ and $\theta = 0.3$, then $h(0.3) = 0.881$.

Now $E[L(X_1^N)] = 4(0.027 + 0.063 + 0.063 + 0.063) + 3(0.147 + 0.147) + 2(0.147 + 0.343) = 2.726$. Therefore $R = 2.726/3 = 0.909$. 

Frans M.J. Willems

INTRODUCTION

HUFFMAN-TUNSTALL
Binary IID Sources
Huffman Code
Tunstall Code

ENUMERATIVE CODING
Lexicographical Ordering
FV: Pascal-\(\Delta\) Method
VF: Petry Code

ARITHMETIC CODING
Intervals
Universal Coding, Individual Redundancy

CONTEXT-TREE
WEIGHTING
IID, unknown \(\theta\)
Tree Sources
Context Trees
Coding Prbs., Redundancy

REPETITION TIMES
LZ77
Repetition Times, Kac
Repetition-Time Algorithm
Achieving Entropy
Let $N = 3$ and $\theta = 0.3$, then $h(0.3) = 0.881$. 

Now $E[L(X_1^N)] = 4(.027 + .063 + .063 + .063) + 3(.147 + .147) + 2(.147 + .343) = 2.726$. Therefore $R = 2.726/3 = 0.909$. 
Let $N = 3$ and $\theta = 0.3$, then $h(0.3) = 0.881$.

Now $E[L(X_1^N)] = 4(.027 + .063 + .063 + .063) + 3(.147 + .147) + 2(.147 + .343) = 2.726$. Therefore $R = 2.726/3 = 0.909$. 
Huffman’s Construction

Example

Let $N = 3$ and $\theta = 0.3$, then $h(0.3) = 0.881$.

Now $E[L(X^N_1)] = 4(0.027 + 0.063 + 0.063 + 0.063) + 3(0.147 + 0.147) + 2(0.147 + 0.343) = 2.726$. Therefore $R = 2.726/3 = 0.909$. 
Huffman’s Construction

Example

Let \( N = 3 \) and \( \theta = 0.3 \), then \( h(0.3) = 0.881 \).

Now \( E[L(X_1^N)] = 4(0.027 + 0.063 + 0.063 + 0.063) + 3(0.147 + 0.147) + 2(0.147 + 0.343) = 2.726 \). Therefore \( R = 2.726/3 = 0.909 \).
Huffman’s Construction

Example

Let $N = 3$ and $\theta = 0.3$, then $h(0.3) = 0.881$.

Now $E[L(X_1^N)] = 4(0.027 + 0.063 + 0.063 + 0.063) + 3(0.147 + 0.147) + 2(0.147 + 0.343) = 2.726$. Therefore $R = 2.726/3 = 0.909$. 
Remarks: Huffman Code

- Note that \( R \downarrow h(\theta) \) when \( N \to \infty \).
- Always \( E[L(X_1^N)] \geq 1 \). For \( \theta \approx 0 \) a Huffman code has expected codeword length \( E[L(X_1^N)] \approx 1 \) and rate \( R \approx 1/N \).
- Better bounds exist for Huffman codes than \( E[L(X_1^N)] < H(X_1^N) + 1 \). E.g. Gallager [1978] showed that
  \[
  E[L(X_1^N)] - H(X_1^N) \leq \max_{X_1^N} P(x_1^N) + 0.086.
  \]
- Adaptive Huffman Codes (Gallager [1978]).
Variable-to-Fixed (VF) Length Codes

**IDEA:**

Parse the source output into variable-length segments of roughly the same probability. Code all these segments with codewords of fixed length.

**Definition (VF-LENGTH Code):**

A **VF-length code** is defined by a set of variable-length source segments. Each segment \( x^* \) in the set gets a unique binary codeword \( c(x^*) \) of length \( L \). The length of a segment \( x^* \) is denoted as \( N(x^*) \). The rate of a VF-code is

\[
R \triangleq \frac{L}{E[N(X^*)]} \quad \text{(code-symbols/source symbol)}.
\]

**GOAL:**

We would like to find parsable VF-length codes that **MINIMIZE** this rate.
Proper-and-Complete Segment Sets

**Definition (Proper-and-Complete Segment Sets)**

A set of source segments is proper-and-complete if each semi-infinite source sequence has a unique prefix in this segment set.

We focus on proper-and-complete segments sets. Segments in a proper-and-complete set can be regarded as leaves in a rooted tree. Such sets guarantee instantaneous parsability.

**Example**

<table>
<thead>
<tr>
<th>$x^*$</th>
<th>$N(x^*)$</th>
<th>$c(x^*)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>01</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>001</td>
<td>3</td>
<td>01</td>
</tr>
<tr>
<td>000</td>
<td>3</td>
<td>00</td>
</tr>
</tbody>
</table>
Assume that the source is IID with parameter $\theta$. Consider a set of segments and all their prefixes. Depict them in a tree. The segments are **leaves**, the prefixes **nodes**. Note that all the nodes and leaves have a probability. E.g. $P(10) = \theta(1 - \theta)$. Let $F(\cdot)$ be a function on nodes, leaves.

**Lemma (Massey, 1983)**

$$\sum_{l \in \text{leaves}} P(l) [F(l) - F(\emptyset)] = \sum_{n \in \text{nodes}} P(n) \sum_{s \in \text{sons of } n} \frac{P(s)}{P(n)} [F(s) - F(n)].$$

- Let $F(x^*) = \#$ of edges from $x^*$ to root, then
  $$E[N(X^*)] = \sum_{x^* \in \text{nodes}} P(x^*).$$
- Let $F(x^*) = -\log_2 P(x^*)$, then
  $$H(X^*) = E[N(X^*)] h(\theta).$$
Assume that the source is IID with parameter $\theta$. Consider a set of segments and all their prefixes. Depict them in a tree. The segments are leaves, the prefixes nodes. Note that all the nodes and leaves have a probability. E.g. $P(10) = \theta(1 - \theta)$. Let $F(\cdot)$ be a function on nodes, leaves.

$$
\begin{align*}
\sum_{l \in \text{leaves}} P(l)[F(l) - F(\emptyset)] &= \sum_{n \in \text{nodes}} P(n) \sum_{s \in \text{sons of } n} \frac{P(s)}{P(n)}[F(s) - F(n)].
\end{align*}
$$

- Let $F(x^*) = \#$ of edges from $x^*$ to root, then
  $$
  E[N(X^*)] = \sum_{x^* \in \text{nodes}} P(x^*).
  $$

- Let $F(x^*) = -\log_2 P(x^*)$, then
  $$
  H(X^*) = E[N(X^*)] h(\theta).
  $$
Proper-and-Complete Segment Sets: Result

**Theorem**

For any proper-and-complete segment set with no more than $2^L$ segments

$$L \geq H(X^*) = E[N(X^*)]h(\theta),$$

or

$$R = \frac{L}{E[N(X^*)]} \geq h(\theta).$$

More precisely, since

$$R = \frac{L}{E[N(X^*)]} = \frac{L}{H(X^*)}h(\theta),$$

we should make $H(X^*)$ as close as possible to $L$, hence **all segments should have roughly the same probability.**
Consider $0 < \theta \leq 1/2$.

**Definition (Optimal VF-length Code)**

A code that maximizes the expected segment-length $E[N(X^*)]$ is called **optimal**. Such a code minimizes the rate $R$.

**Theorem (Tunstall, 1967)**

The Tunstall construction leads to an optimal code.

**CONSTRUCTION:**

- Start with the empty segment $\emptyset$ which has unit probability.
- As long as the number of segments is smaller than $2^L$ replace a segment $s$ with largest probability $P(s)$ by two segments $s_0$ and $s_1$. The probabilities of the new segments (leaves) are $P(s_0) = P(s)(1 - \theta)$ and $P(s_1) = P(s)\theta$.

The Tunstall construction results in $H(X^*) \geq L - \log_2(1/\theta)$ and therefore $R \leq \frac{L}{L + \log_2(\theta)} h(\theta)$ (Jelinek and Schneider [1972]).
Tunstall’s Code

Consider $0 < \theta \leq 1/2$.

Definition (Optimal VF-length Code)

A code that maximizes the expected segment-length $E[N(X^*)]$ is called optimal. Such a code minimizes the rate $R$.

Theorem (Tunstall, 1967)

The Tunstall construction leads to an optimal code.

CONSTRUCTION:

- Start with the empty segment $\emptyset$ which has unit probability.
- As long as the number of segments is smaller than $2^L$ replace a segment $s$ with largest probability $P(s)$ by two segments $s_0$ and $s_1$. The probabilities of the new segments (leaves) are $P(s_0) = P(s)(1 - \theta)$ and $P(s_1) = P(s)\theta$.

The Tunstall construction results in $H(X^*) \geq L - \log_2(1/\theta)$ and therefore $R \leq \frac{L}{L + \log_2(\theta)} h(\theta)$ (Jelinek and Schneider [1972]).
Tunstall’s Construction

Example

Let $L = 3$ and $\theta = 0.3$. Again $h(0.3) = 0.881$.

Now $E[N(X^*)] = 1.0 + 0.7 + 0.3 + 0.49 + 0.21 + 0.343 + 0.240 = 3.283$ and therefore $R = 3/3.283 = 0.914$. 
Example

Let \( L = 3 \) and \( \theta = 0.3 \). Again \( h(0.3) = 0.881 \).

Now \( E[N(X^*)] = 1.0 + 0.7 + 0.3 + 0.49 + 0.21 + 0.343 + 0.240 = 3.283 \) and therefore \( R = 3/3.283 = 0.914 \).
Example

Let $L = 3$ and $\theta = 0.3$. Again $h(0.3) = 0.881$.

\[
\text{Now } E[N(X^*)] = 1.0 + .7 + .3 + .49 + .21 + .343 + .240 = 3.283 \text{ and therefore } R = \frac{3}{3.283} = 0.914.
\]
Example

Let $L = 3$ and $\theta = 0.3$. Again $h(0.3) = 0.881$.

Now $E[N(X^*)] = 1.0 + .7 + .3 + .49 + .21 + .343 + .240 = 3.283$ and therefore $R = 3/3.283 = 0.914$. 
Let \( L = 3 \) and \( \theta = 0.3 \). Again \( h(0.3) = 0.881 \).

Now \( E[N(X^*)] = 1.0 + 0.7 + 0.3 + 0.49 + 0.21 + 0.343 + 0.240 = 3.283 \) and therefore \( R = 3/3.283 = 0.914 \).
Example

Let \( L = 3 \) and \( \theta = 0.3 \). Again \( h(0.3) = 0.881 \).

Now \( E[N(X^*]) = 1.0 + 0.7 + 0.3 + 0.49 + 0.21 + 0.343 + 0.240 = 3.283 \) and therefore \( R = 3/3.283 = 0.914 \).
Let $L = 3$ and $\theta = 0.3$. Again $h(0.3) = 0.881$.

Now $E[N(X^*)] = 1.0 + 0.7 + 0.3 + 0.49 + 0.21 + 0.343 + 0.240 = 3.283$ and therefore $R = 3/3.283 = 0.914$. 
Tunstall’s Construction

Example

Let $L = 3$ and $\theta = 0.3$. Again $h(0.3) = 0.881$.

Now $E[N(X^*)] = 1.0 + 0.7 + 0.3 + 0.49 + 0.21 + 0.343 + 0.240 = 3.283$ and therefore $R = 3/3.283 = 0.914$. 
Tunstall’s Construction

Example

Let \( L = 3 \) and \( \theta = 0.3 \). Again \( h(0.3) = 0.881 \).

Now \( E[N(X^*)] = 1.0 + .7 + .3 + .49 + .21 + .343 + .240 = 3.283 \) and therefore \( R = 3/3.283 = 0.914 \).
Remarks: Tunstall Code

- Note that $R \downarrow h(\theta)$ when $L \to \infty$.
- For $\theta \approx 0$ a Tunstall code has expected segment length $E[N(X^*)] \approx 2^L - 1$ and rate $R \approx L/(2^L - 1)$. Better than Huffman for $L = N$.
- In each step in the Tunstall procedure, a leaf with the largest probability is changed into a node. This leads to:
  - The largest increase in expected segment length (Massey LN-lemma),
  - and $P(n) \geq P(l)$ for all nodes $n$ and leaves $l$.
  - Therefore for any two leaves $l$ and $l'$ we can say that
    $$P(l) \geq \theta P(n) \geq P(l').$$

  So leaves cannot differ too much in probability. This fact is used to lower bound $H(X^*) \geq L - \log_2(1/\theta)$.
- Optimal VF-length codes can also be found by fixing a number $\gamma$ and defining a node to be internal if its probability is $\geq \gamma$ (Khodak, 1969). The size of the segment set is not completely controllable now.
- Run-length Codes (Golomb [1966]).
INTRODUCTION

1. HUFFMAN and TUNSTALL
   - Binary IID Sources
   - Huffman Code
   - Tunstall Code

2. ENUMERATIVE CODING
   - Lexicographical Ordering
   - FV: Pascal-Δ Method
   - VF: Petry Code

3. ARITHMETIC CODING
   - Intervals
   - Universal Coding, Individual Redundancy

4. CONTEXT-TREE WEIGHTING
   - IID, unknown $\theta$
   - Binary Tree-Sources
   - Context Trees
   - Coding Probabilities

5. REPETITION TIMES
   - LZ77
   - Repetition Times, Kac
   - Repetition-Time Algorithm
   - Achieving Entropy

6. CONCLUSION
Lexicographical Ordering

IDEA:

Sequences having the same weight (and probability) only need to be INDEXED. The binary representation of the index can be taken as codeword.

Definition (Lexicographical Ordering, Index)

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 \(S = \{x_1^N : w(x_1^N) = 2\}\) where \(w(x_1^N)\) is the weight of \(x_1^N\). Then \(|S| = \binom{5}{2} = 10\) and:

\[
\begin{align*}
i_S(11000) &= 9 & i_S(01100) &= 4 \\
i_S(10100) &= 8 & i_S(01010) &= 3 \\
i_S(10010) &= 7 & i_S(01001) &= 2 \\
i_S(10001) &= 6 & i_S(00110) &= 1 \\
i_S(01100) &= 5 & i_S(00011) &= 0
\end{align*}
\]
Theorem (Cover, 1973)

- From the sequence $x_1^N \in S$ we can compute index

$$i_S(x_1^N) = \sum_{n=1,N: x_n=1} \#S(x_1, x_2, \cdots, x_{n-1}, 0),$$

where $\#S(x_1, x_2, \cdots, x_k)$ denotes the number of sequences in $S$ having prefix $x_1, \cdots, x_k$.

- Moreover from the index $i_S(x_1^N)$ the sequence $x_1^N$ can be computed if numbers $\#S(x_1, x_2, \cdots, x_{n-1}, 0)$ for $n = 1, N$ are available.

The index of a sequence can be represented by a codeword of fixed length $\lceil \log_2 |S| \rceil$.

Example

Index $i_S(10100) = \#S(0) + \#S(100) = \binom{4}{2} + \binom{2}{1} = 6 + 2 = 8$ hence, since $|S| = 10$ the corresponding codeword is 1000.
IDEA:

Index sequences of fixed weight. Later use a Huffman code (or a fixed-length code) to describe the weights.

Example (Lynch (1966), Davisson (1966), Schalkwijk (1972))

Let $N = 5$ and $S = \{x_1^N : \sum x_n = 2\}$. Then $|S| = \binom{5}{2} = 10$.

- **Index from Sequence:**
  
  $i(10100) = 6 + 2 = 8$.

- **Sequence from Index:**
  
  Index $i = 8$, now
  
  a) $8 \geq 6$ hence $x_1 = 1$,
  
  b) $i < 6 + 3$ hence $x_2 = 0$,
  
  c) $i \geq 6 + 2$ hence $x_3 = 1$,
  
  d) $x_4 = x_5 = 0$.

- **Pascal Triangle.**
First note that

\[ H(X_1^N) = H(X_1^N, w(X_1^N)) = H(W) + H(X_1^N | W). \]

If we use enumerative coding for \( X_1^N \) given weight \( w \), since all sequences with a fixed weight have equal probability

\[ E[L(X_1^N | W)] = \sum_{w=0,1,N} P(w) \log_2 \left( \binom{N}{w} \right) \]

\[ < \sum_{w=0,1,N} P(w) \log_2 \binom{N}{w} + 1 = H(X_1^N | W) + 1. \]

If \( W \) is encoded using a Huffman code we obtain

\[ E[L(X_1^N)] = E[L(W)] + E[L(X_1^N | W)] \]

\[ \leq H(W) + 1 + H(X_1^N | W) + 1 \]

\[ = H(X_1^N) + 2. \]

Worse than Huffman, but no big code-table needed however.
Remarks: FV Pascal-Triangle Method

- Enumeration for sequences generated by Markov Sources (Cover [1973]).
- Universal approach:

**Davisson [1966]**

If $W$ is encoded with a fixed-length codeword of $\lceil \log_2(N + 1) \rceil$ bits, then entropy is achieved for every $\theta$ for $N \to \infty$.

- Lexicographical ordering also possible for variable-length source segments.
IDEA:

Modify the Tunstall segment sets such that the segments can be indexed.

Again let $0 < \theta \leq 1/2$. It can be shown that a proper-and-complete segment set is a Tunstall set (maximal $E[N(X^*)]$ given the number of segments) if and only if for all nodes $n$ and all leaves $l$

$$P(n) \geq P(l).$$

Consequence

If the segments $x^*$ in a proper-and-complete segment set satisfy

$$P(x^* - 1) > \gamma \geq P(x^*),$$

where $x^* - 1$ is $x^*$ without the last symbol, this segment set is a Tunstall set. Constant $\gamma$ determines the size of the set.

Since

$$P(x^*) = (1 - \theta)^{n_0(x^*)} \theta^{n_1(x^*)},$$

where $n_0(x^*)$ is the number or zeros in $x^*$ and $n_1(x^*)$ the number of ones in $x^*$, etc., this is equivalent to

$$An_0(x^* - 1) + Bn_1(x^* - 1) < C \leq An_0(x^*) + Bn_1(x^*)$$

for $A = -\log_b(1 - \theta), B = -\log_b \theta, C = -\log_b \gamma$, and some log-base $b$. 
VF: Petry Code (cont.)

Note that log-base $b$ has to satisfy

$$1 = (1 - \theta) + \theta = b^{-A} + b^{-B}.$$ 

For special values of $\theta$, $A$ and $B$ are integers. E.g. for $\theta = (1 - \theta)^2$ we obtain that $A = 1$ and $B = 2$ for $b = (1 + \sqrt{5})/2$. Now $C$ can also assumed to be integer. The corresponding codes are called Petry codes.

**Definition (Petry (Schalkwijk), 1982)**

Fix integers $A$ and $B$. The segments $x^*$ in a proper-and-complete Petry segment set satisfy

$$A n_0(x^{* - 1}) + B n_1(x^{* - 1}) < C \leq A n_0(x^*) + B n_1(x^*).$$

Integer $C$ can be chosen to control the size of the set.

**Linear Array**

Petry codes can be implemented using a linear array.
**Example**


For given $C$, let $S(C)$ denote the resulting segment set and $\sigma(C)$ its cardinality. Let $S(-1) = S(0) = \emptyset$, then $S(1) = \{0, 1\}$, $S(2) = \{00, 01, 1\}$, etc. Moreover now $\sigma(-1) = \sigma(0) = 1$, $\sigma(1) = 2$ and $\sigma(2) = 3$, etc. It is easy to see that

$$\sigma(C) = \sigma(C - 1) + \sigma(C - 2),$$

and therefore $\sigma(3) = 5$, $\sigma(4) = 8$, $\sigma(5) = 13$, $\sigma(6) = 21$, $\sigma(7) = 34$, and $\sigma(8) = 55$.

Now take $C = 8$. Note that $010010 \in S(8)$.

We can now determine the index $i(010010)$ using Cover’s formula:

$$i(010010) = \#(00) + \#(01000) = \sigma(6) + \sigma(2) = 21 + 3 = 24.$$
VF: Petry Code (cont.)

Theorem (Tjalkens & W. (1987))

A Petry code with parameters $A < B$, and $C$ is a Tunstall code for parameter $q$ where $q = b^{-B}$ when $b$ is the solution of $b^{-A} + b^{-B} = 1$. For arbitrary $\theta$ the rate

\[
\frac{\log_2 \sigma(C)}{E[N(X^*)]} \leq \frac{C + (B - 1)}{C} (h(\theta) + d(\theta\|q)).
\]

Example

In the table $q$ for several values of $A$ and $B$:

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>0.382</td>
<td>0.318</td>
<td>0.276</td>
<td>0.245</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>0.4302</td>
<td>0.382</td>
<td>0.346</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>0.450</td>
<td>0.412</td>
<td></td>
</tr>
</tbody>
</table>
Remarks: VF Petry Code

- Note that $\log_2 \sigma(C)/E[N(X^*)] \downarrow h(\theta) + d(\theta|q)$ when $C \to \infty$, hence a Petry code achieves entropy for $\theta = q$.
- Tjalkens and W. investigated VF-length Petry codes for Markov sources, again with a linear array for each state.
- VF-length universal enumerative solutions exist (Lawrence [1977], Tjalkens and W. [1992]).
- The numbers in the linear array show exponential behaviour. Also an array $[2^{-i/M}]_f$ for $i = 1, M$ can be used, through which we make steps (Tjalkens [PhD, 1987]). This reduces the storage complexity and is similar to Rissanen [1976] multiplication-avoiding arithmetic coding (Generalized Kraft inequality).
1 INTRODUCTION

2 HUFFMAN and TUNSTALL
   - Binary IID Sources
   - Huffman Code
   - Tunstall Code

3 ENUMERATIVE CODING
   - Lexicographical Ordering
   - FV: Pascal-∆ Method
   - VF: Petry Code

4 ARITHMETIC CODING
   - Intervals
   - Universal Coding, Individual Redundancy

5 CONTEXT-TREE WEIGHTING
   - IID, unknown $\theta$
   - Binary Tree-Sources
   - Context Trees
   - Coding Probabilities

6 REPETITION TIMES
   - LZ77
   - Repetition Times, Kac
   - Repetition-Time Algorithm
   - Achieving Entropy

7 CONCLUSION
Idea Elias

Elias:

If source sequences are ORDERED LEXICOGRAPHICALLY then codewords can be COMPUTED SEQUENTIALLY from the source sequence using conditional PROBABILITIES of next symbol given the previous ones, and vice versa.
Source Intervals

Definition

Order the source sequences \( x_1^N \in \{0,1\}^N \) lexicographically according to \( 0 < 1 \).

Now, to each source sequence \( x_1^N \in \{0,1\}^N \) there corresponds a source-interval

\[
I(x_1^N) = [Q(x_1^N), Q(x_1^N) + P(x_1^N))
\]

with

\[
Q(x_1^N) = \sum_{\tilde{x}_1^N < x_1^N} P(\tilde{x}_1^N).
\]

By construction the source intervals are all disjoint. Their union is \([0, 1)\).

Example

Consider an I.I.D. source with \( \theta = 0.2 \) and \( N = 2 \).

<table>
<thead>
<tr>
<th>( x_1^N )</th>
<th>( P(x_1^N) )</th>
<th>( Q(x_1^N) )</th>
<th>( I(x_1^N) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0.64</td>
<td>0</td>
<td>([0, 0.64))</td>
</tr>
<tr>
<td>01</td>
<td>0.16</td>
<td>0.64</td>
<td>([0.64, 0.8))</td>
</tr>
<tr>
<td>10</td>
<td>0.16</td>
<td>0.8</td>
<td>([0.8, 0.96))</td>
</tr>
<tr>
<td>11</td>
<td>0.04</td>
<td>0.96</td>
<td>([0.96, 1))</td>
</tr>
</tbody>
</table>
A codeword $c$ with length $L$ can be regarded as a binary fraction $c$. If we concatenate this codeword with others the corresponding fraction can increase, but no more than $2^{-L}$.

**Definition**

To a codeword $c(x_1^N)$ with length $L(x_1^N)$ there corresponds a code interval

$$J(x_1^N) = [c(x_1^N), c(x_1^N) + 2^{-L(x_1^N)}].$$

Note that $J(x_1^N) \subset [0, 1)$. 
Arithmetic coding: Encoding and Decoding

**Procedure**

- **ENCODING:** Choose $c$ such that the code interval $\subseteq$ source interval, i.e.
  
  
  $$[.c, .c + 2^{-L}) \subseteq [Q(x_1^N), Q(x_1^N) + P(x_1^N)).$$

- **DECODING:** Is possible since there is only one source interval that contains the code interval.

**Theorem**

*For sequence $x_1^N$ with source-interval $I(x_1^N) = [Q(x_1^N), Q(x_1^N) + P(x_1^N))$ take $c(x_1^N)$ as the codeword with*

$$L(x_1^N) \triangleq \left\lfloor \log_2 \frac{1}{P(x_1^N)} \right\rfloor + 1$$

$$c(x_1^N) \triangleq \left\lfloor Q(x_1^N) \cdot 2^{L(x_1^N)} \right\rfloor \cdot 2^{-L(x_1^N)}.$$

*Then*

$$J(c(x_1^N)) \subseteq I(x_1^N).$$

*and*

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

*i.e. less than two bits above the ideal codeword length.*
Example

I.I.D. source with $\theta = 0.2$ and $N = 2$.

Source Intervals

Code Intervals

Source-intervals are disjoint $\Rightarrow$ code-intervals are disjoint $\Rightarrow$ prefix condition holds.
Example

I.I.D. source with $\theta = 0.2$ and $N = 2$.

<table>
<thead>
<tr>
<th>Source Intervals</th>
<th>Code Intervals</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>0</td>
</tr>
<tr>
<td>0.96</td>
<td>11</td>
</tr>
<tr>
<td>0.80</td>
<td>10</td>
</tr>
<tr>
<td>0.64</td>
<td>01</td>
</tr>
<tr>
<td>0.00</td>
<td>00</td>
</tr>
</tbody>
</table>

Source-intervals are disjoint $\Rightarrow$ code-intervals are disjoint $\Rightarrow$ prefix condition holds.
Example (Connection to Cover’s formula)

Let $L = 3$ and $\theta = 0.2$.

\[
Q(101) = P(0) + P(100) = 0.8 + 0.2 \cdot 0.8 \cdot 0.8 = 0.928.
\]

\[
P(101) = P(1)P(0)P(1) = 0.2 \cdot 0.8 \cdot 0.2 = 0.032.
\]
Arithmetic Coding: Sequential Computation (Elias)

In general

$$Q(x_1^N) = \sum_{n=1, N: x_n=1} P(x_1, x_2, \cdots, x_{n-1}, 0),$$

$$P(x_1^N) = \prod_{n=1}^{N} P(x_n | x_1, x_2, \cdots, x_{n-1}).$$

Sequential Computation

If we have access to $P(x_1, x_2, \cdots, x_n, 0)$ and $P(x_1, x_2, \cdots, x_n, 1)$ after having processed $P(x_1, x_2, \cdots, x_n)$ for $n = 1, 2, \cdots, N$ we can compute $I(x_1^N)$ sequentially.
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

$$P_c(x_1^N) > 0 \text{ for all } x_1^N,$$

and

$$\sum_{x_1^N} P_c(x_1^N) = 1.$$ 

Then

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

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

$$P_c(x_1^N) > 0 \text{ for all } x_1^N,$$

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 $P_c(x_1^N)$?
**Individual 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.

**Bound Individual Redundancy**

Arithmetic coding based on coding probabilities \( \{ P_c(x_1^N), x_1^N \in \{0,1\}^N \} \) yields

\[
\rho(x_1^N) < \log_2 \frac{1}{P_c(x_1^N)} + 2 - \log_2 \frac{1}{P(x_1^N)} = \log_2 \frac{P(x_1^N)}{P_c(x_1^N)} + 2.
\]

We say that the Coding redundancy \(< 2\) bits.

The coding probabilities should be as large as possible (as close as possible to the actual probabilities). Next focus on remaining part of the individual redundancy

\[
\log_2 \frac{P(x_1^N)}{P_c(x_1^N)}.
\]
• Shannon [1948] already described relation between codewords and intervals, ordered probabilities however. Called Shannon-Fano code.
• Shannon-Fano-Elias, arbitrary ordering, but not sequential.
• Finite precision issues arithmetic coding solved by Pasco [1976] and Rissanen [1976].
INTRODUCTION

HUFFMAN and TUNSTALL
- Binary IID Sources
- Huffman Code
- Tunstall Code

ENUMERATIVE CODING
- Lexicographical Ordering
- FV: Pascal-∆ Method
- VF: Petry Code

ARITHMETIC CODING
- Intervals
- Universal Coding, Individual Redundancy

CONTEXT-TREE WEIGHTING
- IID, unknown θ
- Binary Tree-Sources
- Context Trees
- Coding Probabilities

REPETITION TIMES
- LZ77
- Repetition Times, Kac
- Repetition-Time Algorithm
- Achieving Entropy

CONCLUSION
CTW: Universal Codes

IDEA:
Find good coding probabilities for sources with UNKNOWN PARAMETERS and STRUCTURE. Use WEIGHTING!
Coding for a Binary IID Source, Unknown $\theta$

**Definition (Krichevsky-Trofimov estimator (1981))**

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

$$P_e(a, b) = \int_{\theta=0}^{1} \frac{1}{\pi \sqrt{(1 - \theta)\theta}} \cdot (1 - \theta)^a \theta^b d\theta.$$  

( Dirichlet-$(1/2, 1/2)$ prior, “weighting”).

**Theorem**

- **Upperbound on the PARAMETER redundancy**

  $$\log_2 \frac{P(x_1^N)}{P_c(x_1^N)} = \log_2 \frac{\theta^a(1 - \theta)^b}{P_e(a, b)} \leq \frac{1}{2} \log_2(a + b) + 1 = \frac{1}{2} \log_2(N) + 1.$$  

  for all $\theta$ and $x_1^N$ with $a$ zeros and $b$ ones.

- **Probability of a sequence with $a$ zeroes and $b$ ones followed by a zero**

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

  hence **SEQUENTIAL COMPUTATION is possible!**
The total individual redundancy

\[ \rho(x_1^N) < \log_2 \frac{\theta^a (1 - \theta)^b}{P_e(a, b)} + 2 \leq \left( \frac{1}{2} \log_2(N) + 1 \right) + 2. \]

for all \( \theta \) and \( x_1^N \) with \( a \) zeroes and \( b \) ones.

- Shtarkov [1988]: \( \frac{1}{2} \log_2 N \) behaviour is asymptotically optimal for individual redundancy for \( N \to \infty \) (NML-estimator)!
- Rissanen [1984]: Also for expected redundancy \( \frac{1}{2} \log_2 N \) behaviour is asymptotically optimal.
Definition

\[ \ldots \quad X_{n-2} \quad X_{n-1} \quad X_n \]

\[ \theta_1 = 0.1 \]

\[ \theta_{10} = 0.3 \]

\[ \theta_{00} = 0.5 \]

\begin{align*}
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
\end{align*}

\( (\text{tree-}) \) model \( \mathcal{M} = \{00, 10, 1\} \)
CTW: Binary Tree-Sources

\[
M = \{00, 10, 1\}
\]

\[
P(X_n = 1|\cdots, X_{n-1} = 1) = 0.1
\]
\[
P(X_n = 1|\cdots, X_{n-2} = 1, X_{n-1} = 0) = 0.3
\]
\[
P(X_n = 1|\cdots, X_{n-2} = 0, X_{n-1} = 0) = 0.5
\]
CTW: Binary Tree-Sources

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

\[ P(X_n = 1|\cdots, X_{n-1} = 1) = 0.1 \]
\[ P(X_n = 1|\cdots, X_{n-2} = 1, X_{n-1} = 0) = 0.3 \]
\[ P(X_n = 1|\cdots, X_{n-2} = 0, X_{n-1} = 0) = 0.5 \]
CTW: Problem, Concepts

**PROBLEM:**
What are good coding probabilities for sequences $x_1^N$ produced by a tree-source with

- an unknown tree-model,
- and unknown parameters?

**CONCEPTS:**

- **CONTEXT TREE** (Rissanen [1983])
- **WEIGHTING:** If $P_1(x)$ or $P_2(x)$ are two alternative coding probabilities for sequence $x$, then the **weighted probability**

$$P_w(x) \triangleq \frac{P_1(x) + P_2(x)}{2} \geq \frac{1}{2} \max(P_1(x), P_2(x)),$$

thus we lose at most a factor of 2, which is one bit in redundancy.
CTW: Problem, Concepts

PROBLEM:
What are good coding probabilities for sequences $x_1^n$ produced by a tree-source with
- an unknown tree-model,
- and unknown parameters?

CONCEPTS:
- CONTEXT TREE (Rissanen [1983])
- WEIGHTING: If $P_1(x)$ or $P_2(x)$ are two alternative coding probabilities for sequence $x$, then the weighted probability

$$P_w(x) \triangleq \frac{P_1(x) + P_2(x)}{2} \geq \frac{1}{2} \max(P_1(x), P_2(x)),$$

thus we loose at most a factor of 2, which is one bit in redundancy.
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
The subsequence corresponding to a leaf $s$ of the context tree is IID. A good coding probability for this subsequence is therefore

$$P_w(s) \triangleq P_e(a_s, b_s),$$

where $a_s$ and $b_s$ are the number of zeroes and ones of this subsequence.
Coding Probabilities: Internal nodes of the Context-Tree

The subsequence corresponding to a node $s$ of the context tree is
- IID if the node $s$ is not an internal node of the actual tree-model,
- a combination of the subsequences that correspond to nodes 0$s$ and 1$s$, if $s$ is an internal node of the actual tree-model.
Weighing

CTW: Internal Nodes

Weighting the coding probabilities corresponding to both alternatives yields the coding probability

\[ P_w(s) \triangleq \frac{P_e(a_s, b_s) + P_w(0s) \cdot P_w(1s)}{2} \]

for the subsequence that corresponds to node \( s \).

Recursively we find in the root \( \emptyset \) of the context-tree the coding probability \( P_w(\emptyset) \) for the entire source sequence \( x_1^N \).

IMPORTANT:

Coding probability \( P_w(\lambda) \) can be computed sequentially.
Redundancy (tree-model $\mathcal{M} = \{00, 10, 1\}$)

Actual probability:

$$P(x_1^N) = (1 - \theta_{00})^{a_{00}} \theta_{00}^{b_{00}} (1 - \theta_{10})^{a_{10}} \theta_{10}^{b_{10}} (1 - \theta_1)^{a_1} \theta_1^{b_1}.$$  

Lower bound coding probability:

$$P_w(\emptyset) \geq \frac{1}{2} P_w(0) \cdot P_w(1)$$

$$\geq \frac{1}{2} \frac{1}{2} P_w(00) \cdot P_w(10) \cdot \frac{1}{2} P_e(a_1, b_1)$$

$$\geq \frac{1}{2} \frac{1}{2} \frac{1}{2} P_e(a_{00}, b_{00}) \cdot \frac{1}{2} P_e(a_{10}, b_{10}) \cdot \frac{1}{2} P_e(a_1, b_1).$$

Parameter redundancy bounds for the subsequences in the leaves of tree-model $\mathcal{M} = \{00, 10, 1\}$:

$$\log_2 \frac{(1 - \theta_{00})^{a_{00}} \theta_{00}^{b_{00}}}{P_e(a_{00}, b_{00})} \leq \frac{1}{2} \log_2 (a_{00} + b_{00}) + 1,$$

$$\log_2 \frac{(1 - \theta_{10})^{a_{10}} \theta_{10}^{b_{10}}}{P_e(a_{10}, b_{10})} \leq \frac{1}{2} \log_2 (a_{10} + b_{10}) + 1,$$

$$\log_2 \frac{(1 - \theta_1)^{a_1} \theta_1^{b_1}}{P_e(a_1, b_1)} \leq \frac{1}{2} \log_2 (a_1 + b_1) + 1.$$
Redundancy (General)

\[ \rho(x_1^N) < \log_2 \frac{P(x_1^N)}{P_w(\emptyset)} + 2 \]
\[ \leq \log_2 32 + \frac{1}{2} \log_2 (a_{00} + b_{00}) + 1 + \frac{1}{2} \log_2 (a_{10} + b_{10}) + 1 \]
\[ + \frac{1}{2} \log_2 (a_1 + b_1) + 1 + 2 \]
\[ \leq 5 + \left( \frac{3}{2} \log_2 \frac{N}{3} + 3 \right) + 2, \]
for all \( x_1^N \), and all \( \theta_{00}, \theta_{10}, \) and \( \theta_1 \).

**Theorem (W., Shtarkov, and Tjalkens (1995))**

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

\[ \rho(x_1^N) < (2|\mathcal{M}| - 1) + \left( \frac{|\mathcal{M}|}{2} \log_2 \frac{N}{|\mathcal{M}|} + |\mathcal{M}| \right) + 2 \text{ bits.} \]

*(model, parameter, and coding redundancies)*
Simulation: Model plus Parameter Redundancies

Redundencies for the CTW method, but also for methods focussing on $\mathcal{M} = \emptyset$, $\mathcal{M} = \{0, 1\}$, actual model $\mathcal{M} = \{00, 10, 1\}$, $\mathcal{M} = \{0, 01, 11\}$ and $\mathcal{M} = \{00, 10, 01, 11\}$. The CTW method improves over the best model!
CTW implements a “weighting” (Bayes mixture) over all tree-models with depth not exceeding $D$, i.e.

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

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

There is one tree-model of depth 0 (i.e., the IID model). If there are $\#_d$ models of depth not exceeding $d$ then $\#_{d+1} = \#_d^2 + 1$. Therefore $\#_1 = 2$, $\#_2 = 5$, $\#_3 = 26$, $\#_4 = 677$, $\#_5 = 458330$, $\#_6 = 210066388901$, $\#_7 = 4.4128 \cdot 10^{22}$, $\#_8 = 1.9473 \cdot 10^{45}$, 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.
Remarks: Context-Tree Weighting (cont.)

- 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 \textit{stationary ergodic sources}.
- More general context-algorithms (\textit{splitting rules}) were proposed. The context of \( x_n \) \textbf{need not be} \( x_{n-d}, x_{n-d+1}, \ldots, x_{n-1} \).
- A two-pass version (\textit{context-tree maximizing}) exists that finds the best model (MDL) matching to the source sequence. Now

\[
P_m(s) \triangleq \frac{\max[P_e(a_s, b_s), P_m(0s) \cdot P_m(1s)]}{2}.
\]

If a (minimal) tree source with model generates the sequence \( x_1^N \) the maximizing method produces a model estimate which is correct with probability one as \( N \to \infty \).
Lempel-Ziv 1977 Compression

**IDEA:**
Let the data speak for itself.

LZ77 Compression is achieved by replacing repeated segments in the data with pointers and lengths. To avoid deadlock an uncoded symbol is added to each pointer and length.

**Example (LZ77)**

<table>
<thead>
<tr>
<th>search buffer</th>
<th>look-ahead buffer</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>a b r a c a d a b r a -</td>
<td>(0,-,a)</td>
</tr>
<tr>
<td></td>
<td>a b r a c a d a b r a -</td>
<td>(0,-,b)</td>
</tr>
<tr>
<td></td>
<td>a b r a c a d a b r a -</td>
<td>(0,-,r)</td>
</tr>
<tr>
<td></td>
<td>a b r a c a d a b r a -</td>
<td>(3,1,c)</td>
</tr>
<tr>
<td></td>
<td>a b r a c a d a b r a -</td>
<td>(2,1,d)</td>
</tr>
<tr>
<td></td>
<td>a b r a c a d a b r a -</td>
<td>(7,4,_)</td>
</tr>
<tr>
<td>a b r a c a d a b r a -</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**QUESTION:**
Why does this method work? Note that the statistics of the data are unknown!
Lempel-Ziv 1977 Compression

**IDEA:**

Let the data speak for itself.

LZ77 Compression is achieved by replacing repeated segments in the data with pointers and lengths. To avoid deadlock an uncoded symbol is added to each pointer and length.

**Example (LZ77)**

<table>
<thead>
<tr>
<th>search buffer</th>
<th>look-ahead buffer</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>a b r a c a d a b r a -</td>
<td>(0,-,a)</td>
</tr>
<tr>
<td></td>
<td>a b r a c a d a b r a -</td>
<td>(0,-,b)</td>
</tr>
<tr>
<td></td>
<td>a b r a c a d a b r a -</td>
<td>(0,-,r)</td>
</tr>
<tr>
<td></td>
<td>a b r a c a d a b r a -</td>
<td>(3,1,c)</td>
</tr>
<tr>
<td></td>
<td>a b r a c a d a b r a -</td>
<td>(2,1,d)</td>
</tr>
<tr>
<td></td>
<td>a b r a c a d a b r a -</td>
<td>(7,4,-)</td>
</tr>
</tbody>
</table>

**QUESTION:**

Why does this method work? Note that the statistics of the data are unknown!
Repetition Times

Consider the **discrete stationary and ergodic** process

\[ \ldots, X_{-3}, X_{-2}, X_{-1}, X_0, X_1, X_2, \ldots. \]

Suppose that \( X_1 = x \) for symbol-value \( x \in \mathcal{X} \) with \( \Pr\{X_1 = x\} > 0 \). We say that the **repetition time** of the \( x \) that occurred at time \( t = 1 \) is \( m \) if \( X_{1-m} = x \) and \( X_t \neq x \) for \( t = 2 - m, \ldots, 0 \).

\[
\begin{array}{cccc|cc}
& X_{-3} & X_{-2} & X_{-1} & X_0 & X_1 & X_2 \\
\hline
\text{m = 4} & = x & \neq x & \neq x & \neq x & = x
\end{array}
\]

**Definition (Average Repetition Time)**

Let \( Q_x(m) \) be the conditional probability that the repetition time of the \( x \) occurring at \( t = 1 \) is \( m \), hence

\[ Q_x(m) \triangleq \Pr\{X_{1-m} = x, X_{2-m} \neq x, \ldots, X_0 \neq x | X_1 = x\}. \]

The **average** repetition time for symbol-value \( x \) with \( \Pr\{X_1 = x\} > 0 \) is now defined as

\[ T(x) \triangleq \sum_{m=1,2,\ldots} mQ_x(m). \]
Consider the **discrete stationary and ergodic** process

\[
\cdots, X_{-3}, X_{-2}, X_{-1}, X_0, X_1, X_2, \cdots.
\]

Suppose that \(X_1 = x\) for symbol-value \(x \in \mathcal{X}\) with \(\Pr\{X_1 = x\} > 0\). We say that the **repetition time** of the \(x\) that occurred at time \(t = 1\) is \(m\) if \(X_{1-m} = x\) and \(X_t \neq x\) for \(t = 2 - m, \cdots, 0\).

\[
\begin{array}{c|c|c|c|c}
   m = 4 & X_{-3} & X_{-2} & X_{-1} & X_0 \\
   \hline
   \ = x & \neq x & \neq x & \neq x \\
   \ = x
\end{array}
\]

**Definition (Average Repetition Time)**

Let \(Q_x(m)\) be the conditional probability that the repetition time of the \(x\) occurring at \(t = 1\) is \(m\), hence

\[
Q_x(m) \triangleq \Pr\{X_{1-m} = x, X_{2-m} \neq x, \cdots, X_0 \neq x | X_1 = x\}.
\]

The **average** repetition time for symbol-value \(x\) with \(\Pr\{X_1 = x\} > 0\) is now defined as

\[
T(x) \triangleq \sum_{m=1,2,\cdots} mQ_x(m).
\]
Kac’s Result

Example

Consider an IID (binary) process and assume that \( \Pr\{X_1 = 1\} = \theta > 0 \). Then

\[
Q_1(m) = \theta(1 - \theta)^{m-1} \quad \text{and} \quad T(1) = \sum_{m=1,2,\ldots} m\theta(1 - \theta)^{m-1} = \frac{1}{\theta}.
\]

Theorem (Kac, 1947)

For stationary and ergodic processes, for any \( x \) with \( \Pr\{X_1 = x\} > 0 \),

\[
T(x) = \frac{1}{\Pr\{X_1 = x\}}.
\]

Note that Kac’s result holds also for “sliding” \( N \)-blocks, hence

\[
T((x_1, x_2, \ldots, x_N)) = \frac{1}{\Pr\{(X_1, X_2, \ldots, X_N) = (x_1, x_2, \ldots, x_N)\}},
\]

if \( \Pr\{(X_1, X_2, \ldots, X_N) = (x_1, x_2, \ldots, x_N)\} > 0 \). Now the repetition time is equal to \( m \) when \( m \) is the smallest positive integer such that

\[
(x_1-m, x_2-m, \ldots, x_N-m) = (x_1, x_2, \ldots, x_N).
\]
Kac’s Result

Example

Consider an IID (binary) process and assume that \( \Pr\{X_1 = 1\} = \theta > 0 \). Then

\[
Q_1(m) = \theta(1 - \theta)^{m-1} \quad \text{and} \quad T(1) = \sum_{m=1,2,\ldots} m\theta(1 - \theta)^{m-1} = \frac{1}{\theta}.
\]

Theorem (Kac, 1947)

For stationary and ergodic processes, for any \( x \) with \( \Pr\{X_1 = x\} > 0 \),

\[
T(x) = \frac{1}{\Pr\{X_1 = x\}}.
\]

Note that Kac’s result holds also for “sliding” \( N \)-blocks, hence

\[
T((x_1, x_2, \ldots, x_N)) = \frac{1}{\Pr\{X_1, X_2, \ldots, X_N = (x_1, x_2, \ldots, x_N)\}}
\]

if \( \Pr\{X_1, X_2, \ldots, X_N = (x_1, x_2, \ldots, x_N)\} > 0 \). Now the repetition time is equal to \( m \) when \( m \) is the smallest positive integer such that

\((x_{1-m}, x_{2-m}, \ldots, x_{N-m}) = (x_1, x_2, \cdots, x_N)\).
Kac’s Result

Example

Consider an IID (binary) process and assume that $\Pr\{X_1 = 1\} = \theta > 0$. Then

$$Q_1(m) = \theta(1 - \theta)^{m-1} \quad \text{and}$$

$$T(1) = \sum_{m=1,2,\ldots} m\theta(1 - \theta)^{m-1} = \frac{1}{\theta}.$$ 

Theorem (Kac, 1947)

For stationary and ergodic processes, for any $x$ with $\Pr\{X_1 = x\} > 0$,

$$T(x) = \frac{1}{\Pr\{X_1 = x\}}.$$ 

Note that Kac’s result holds also for “sliding” $N$-blocks, hence

$$T((x_1, x_2, \ldots, x_N)) = \frac{1}{\Pr\{(X_1, X_2, \ldots, X_N) = (x_1, x_2, \ldots, x_N)\}},$$

if $\Pr\{(X_1, X_2, \ldots, X_N) = (x_1, x_2, \ldots, x_N)\} > 0$. Now the repetition time is equal to $m$ when $m$ is the smallest positive integer such that

$$(x_1-m, x_2-m, \ldots, x_N-m) = (x_1, x_2, \ldots, x_N).$$
Suppose that our source is binary i.e. $X_t \in \{0, 1\}$ for all integer $t$.

A. The encoder wants to convey a source block $x_1^N \triangleq (x_1, x_2, \cdots, x_N)$ to the decoder. Both encoder and decoder have access to buffers containing all previous source symbols $\cdots, x_{-2}, x_{-1}, x_0$.

B. Using these previous source symbols the encoder can determine the repetition time $m$ of $x_1^N$. It is the smallest integer $m$ that satisfies

$$x_{1-m}^{N-m} = x_1^N,$$

where $x_{1-m}^{N-m} \triangleq (x_1-m, x_2-m, \cdots, x_{N-m})$. 

Suppose that our source is binary i.e. $X_t \in \{0, 1\}$ for all integer $t$. 

\[
\begin{array}{cccccccc}
X_7 & X_6 & X_5 & X_4 & X_3 & X_2 & X_1 & X_0 \\
\Vert & \Vert & \Vert & \Vert & \Vert & \Vert & \Vert \\
X_1 & X_2 & X_3 \\
\end{array}
\]
C. Repetition time \( m \) is now encoded and sent to the decoder. The code for \( m \) consists of a preamble \( p(m) \) and an index \( i(m) \) and has length \( l(m) \).

**Example**

<table>
<thead>
<tr>
<th>( m )</th>
<th>( p(m) )</th>
<th>( i(m) )</th>
<th>( l(m) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>00</td>
<td>-</td>
<td>2+0=2</td>
</tr>
<tr>
<td>2</td>
<td>01</td>
<td>0</td>
<td>2+1=3</td>
</tr>
<tr>
<td>3</td>
<td>01</td>
<td>1</td>
<td>2+1=3</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>00</td>
<td>2+2=4</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>01</td>
<td>2+2=4</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>10</td>
<td>2+2=4</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>11</td>
<td>2+2=4</td>
</tr>
<tr>
<td>( \geq 8 )</td>
<td>11</td>
<td>copy of ( x_1 x_2 x_3 )</td>
<td>2+3=5</td>
</tr>
</tbody>
</table>

**In general** there are \( N + 1 \) groups. There are index groups with 1, 2, up to \( 2^{N-1} \) elements, hence the index lengths are 0, 1, up to \( N - 1 \). The last group is the “copy”-group. A “copy” has length \( N \). We use a preamble \( p(m) \) of \( \lceil \log_2(N + 1) \rceil \) bits to specify one of these \( N + 1 \) alternatives.
For arbitrary \( N \) we get for the code-block length \( l(m) \)

\[
l(m) = \begin{cases} 
  \lceil \log_2(N + 1) \rceil + \lceil \log_2 m \rceil & \text{if } m < 2^N, \\
  \lceil \log_2(N + 1) \rceil + N & \text{if } m \geq 2^N.
\end{cases}
\]

This results in the upper bound

\[
l(m) \leq \lceil \log_2(N + 1) \rceil + \log_2 m.
\]

D. After decoding \( m \) the decoder can reconstruct \( x_1^N \) using the previous source symbols in the buffer. With this block \( x_1^N \) **both the encoder and decoder can update their buffers**.

E. Then the next block

\[X_{N+1}^{2N} \triangleq X_{N+1}, X_{N+2}, \ldots, X_{2N}\]

is processed in a similar way, etc.

**Note:**

Buffers need only contain the previous \( 2^N - 1 \) source symbols!
For arbitrary $N$ we get for the code-block length $l(m)$

$$l(m) = \begin{cases} \lceil \log_2(N + 1) \rceil + \lfloor \log_2 m \rfloor & \text{if } m < 2^N, \\ \lceil \log_2(N + 1) \rceil + N & \text{if } m \geq 2^N. \end{cases}$$

This results in the upper bound

$$l(m) \leq \lceil \log_2(N + 1) \rceil + \log_2 m.$$ 

D. After decoding $m$ the decoder can reconstruct $x_1^N$ using the previous source symbols in the buffer. With this block $x_1^N$ both the encoder and decoder can update their buffers.

E. Then the next block

$$x_{N+1}^{2N} \triangleq x_{N+1}, x_{N+2}, \ldots, x_{2N}$$

is processed in a similar way, etc.

**Note:**

Buffers need only contain the previous $2^N - 1$ source symbols!
Analysis of the Repetition-Time Algorithm

Assume that a certain \( x_1^N \) occurred as first block. What is then the average codeword length \( L(x_1^N) \) for \( x_1^N \)?

\[
L(x_1^N) = \sum_{m=1,2,\ldots} Q_{x_1^N}(m) l(m)
\]

\[
\leq \sum_{m=1,2,\ldots} Q_{x_1^N}(m) \left[ \log_2(N + 1) \right] + \sum_{m=1,2,\ldots} Q_{x_1^N}(m) \log_2 m
\]

\[
\leq \left[ \log_2(N + 1) \right] + \log_2 \left( \sum_{m=1,2,\ldots} m Q_{x_1^N}(m) \right)
\]

\[
\leq \left[ \log_2(N + 1) \right] + \log_2 \left( \frac{1}{\Pr\{X_1^N = x_1^N\}} \right)
\]

Here (a) follows from the upper bound for \( l(m) \), (b) from Jensen’s inequality. Furthermore (c) follows from Kac’s theorem. Ideal codeword length plus \( \left[ \log_2(N + 1) \right] \).
The probability that $x_1^N$ occurs as block is $\Pr\{X_1^N = x_1^N\}$. For the average codeword length $L(X_1^N)$ we therefore get

$$L(X_1^N) = \sum_{x_1^N} \Pr\{X_1^N = x_1^N\} L(x_1^N)$$

$$\leq \sum_{x_1^N} \Pr\{X_1^N = x_1^N\} \left( \lceil \log_2(N + 1) \rceil + \log_2 \frac{1}{\Pr\{X_1^N = x_1^N\}} \right)$$

$$= \lceil \log_2(N + 1) \rceil + H(X_1^N).$$

For the rate $R_N$ we now obtain

$$R_N = \frac{L(X_1^N)}{N} \leq \frac{H(X_1^N)}{N} + \frac{\lceil \log_2(N + 1) \rceil}{N}.$$
The probability that $x_1^N$ occurs as block is $\Pr\{X_1^N = x_1^N\}$. For the average codeword length $L(X_1^N)$ we therefore get

$$L(X_1^N) = \sum_{x_1^N} \Pr\{X_1^N = x_1^N\} L(x_1^N)$$

$$\leq \sum_{x_1^N} \Pr\{X_1^N = x_1^N\} \left( \lceil \log_2(N + 1) \rceil + \log_2 \frac{1}{\Pr\{X_1^N = x_1^N\}} \right)$$

$$= \lceil \log_2(N + 1) \rceil + H(X_1^N).$$

For the rate $R_N$ we now obtain

$$R_N = \frac{L(X_1^N)}{N} \leq \frac{H(X_1^N)}{N} + \frac{\lceil \log_2(N + 1) \rceil}{N}.$$

Theorem (W., 1986, 1989)

The repetition-time algorithm achieves entropy since

$$\lim_{N \to \infty} R_N = \lim_{N \to \infty} \left( \frac{H(X_1^N)}{N} + \frac{\lceil \log_2(N + 1) \rceil}{N} \right) = H_{\infty}(X).$$
Remarks: Repetition-Time Algorithm

- **Universal** algorithm.
- Assume that \( \cdots, X_{-1}, X_0, X_1, X_2, \cdots \) is stationary and ergodic with entropy \( H_\infty(X) \). Let the random variable \( M \) be the repetition time of the source block \( X_1^N \).

**Theorem (Wyner & Ziv, 1989)**

Fix an \( \epsilon > 0 \). Then

\[
\lim_{N \to \infty} \Pr \left\{ M \geq 2^N(H_\infty(X) + \epsilon) \right\} = 0.
\]

- This result implies that the buffer can be much smaller than \( 2^N - 1 \) if the entropy is known to be smaller than 1.
- This result was crucial in proving that the LZ77 algorithm achieves entropy (Wyner & Ziv [1994]).

- Elias [1987], interval and recency-rank coding methods (symbols).
- Hershkovitz and Ziv [1998] studied **conditional repetition times**.

- **When better than CTW?**
  CTW incremental redundancy for an \( N \)-block is \( \approx NK/(2B \ln(2)) \) bits for \( K \) parameters. This redundancy is larger than \( \log_2(N + 1) \) for \( (K/2^N) > 2 \ln(N + 1)/N \). For \( N = 24 \) we get \( K/2^N > 0.2682 \).
INTRODUCTION

HUFFMAN and TUNSTALL
- Binary IID Sources
- Huffman Code
- Tunstall Code

ENUMERATIVE CODING
- Lexicographical Ordering
- FV: Pascal-∆ Method
- VF: Petry Code

ARITHMETIC CODING
- Intervals
- Universal Coding, Individual Redundancy

CONTEXT-TREE WEIGHTING
- IID, unknown $\theta$
- Binary Tree-Sources
- Context Trees
- Coding Probabilities

REPETITION TIMES
- LZ77
- Repetition Times, Kac
- Repetition-Time Algorithm
- Achieving Entropy

CONCLUSION
Recent developments:

- **DUDE** (Weismann, Ordentlich, Serroussi, Verdu, Weinberger [2005]), resulted in study of **bi-directional contexts and splitting rules** (Ordentlich, Weinberger, and Weissman [2005]).

- **Directed Mutual Information** (Marko [1973], Massey [1990]):
  \[
  I(X_1^n \rightarrow Y_1^n) = \sum H(Y_n|Y_1^{n-1}) - H(Y_n|Y_1^{n-1}, X_1^{n-1}, X_n)
  \]
  is a generalisation of Granger causality [1969]. CTW-methods were used to estimate these quantities (Liao, Permuter, Kim, Zhao, Kim, and Weissman [2012]).

Questions:

- LZ learns from seeing once. CTW is optimal for tree sources but seems to take more time. What are the algorithms **between** CTW and LZ?

- Suppose that the data have **left-right symmetry** hence
  \[
  P(a, b) = P(b, a), \quad P(a, b, c) = P(c, b, a), \quad P(a, b, c, d) = P(d, c, b, a),
  \]
  etc. This reduces the number of parameters. Algorithm? Relevant for image-compression.

- CTW can handle side-information by considering it as context (e.g. Cai, Kulkarni and Verdu [2005]). But what if the **side-information is not-properly aligned**? Relevant for reference-based genome compression (Chern et al. [2012]).
Source Coding is FUN!

(Ulm, 1997)