COMP2610 / 6261 — Information Theory

Lecture 13: Symbol Codes for Lossless Compression

Australian National University

17 September, 2018

Last time

Proof of the source coding theorem

- Fou
- Re

The theor

- Could variable length coding help?
- Does entropy turn up for such codes as well?

This time

Variable-length codes

Prefix codes

Kraft's inequality

Codes: A Review

Notation:

- If $\mathcal{A}$ is a finite set then $\mathcal{A}^N$ is the set of all *strings of length N*.
- $\mathcal{A}^+ = \bigcup_N \mathcal{A}^N$ is the set of *all finite strings*

Examples:

- $\{0,$ ^{3}
- $\{0,$

Codes: A Review

Notation:

- If $\mathcal{A}$ is a finite set then $\mathcal{A}^N$ is the set of all *strings of length N*.
- $\mathcal{A}^+ = \bigcup_N \mathcal{A}^N$ is the set of *all finite strings*

Examples:

- $\{0,$ ^{3}
- $\{0,$

Binary S

Let X be an ensemble with $\mathcal{A}_X = \{a_1, \ldots,$
A function $c : \mathcal{A}_X \mapsto \{0,1\}^+$ is a **code** for

- The binary string $c(x)$ is the **codewo**

Codes: A Review

Notation:

- If $\mathcal{A}$ is a finite set then $\mathcal{A}^N$ is the set of all *strings of length N*.
- $\mathcal{A}^+ = \bigcup_N \mathcal{A}^N$ is the set of *all finite strings*

Examples:

- $\{0,$ ^{3}
- $\{0,$

Binary S

Let X be an ensemble with $\mathcal{A}_X = \{a_1, \ldots,$
A function $c : \mathcal{A}_X \mapsto \{0,1\}^+$ is a **code** for

- The binary string $c(x)$ is the **codewo**
- The **length** of the codeword for for x is denoted $\ell(x)$.
  Shorthand: $\ell_i = \ell(a_i)$ for $i = 1 \ldots, I$.

Codes: A Review

Notation:

- If $\mathcal{A}$ is a finite set then $\mathcal{A}^N$ is the set of all *strings of length N*.
- $\mathcal{A}^+ = \bigcup_N \mathcal{A}^N$ is the set of *all finite strings*

Examples:

- $\{0,$ ^{3}
- $\{0,$

Binary S

Let X be an ensemble with $\mathcal{A}_X = \{a_1, \ldots,$
A function $c : \mathcal{A}_X \to \{0,1\}^+$ is a **code** for

- The binary string $c(x)$ is the **codewo**
- The **length** of the codeword for for x is denoted $\ell(x)$.
  Shorthand: $\ell_i = \ell(a_i)$ for $i = 1 \ldots, I$.
- The **extension** of c assigns codewords to any sequence $x_1 x_2 \ldots x_N$ from $\mathcal{A}^+$ by $c(x_1 \ldots x_N) = c(x_1) \ldots c(x_N)$

Codes: A Review

Examples

X is an ensemble with $\mathcal{A}_X = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$

Example 1 (Uniform Code):

Codes: A Review

Examples

X is an ensemble with $\mathcal{A}_X = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$

Example 1 (Uniform Code):

- Let

Codes: A Review

Examples

X is an ensemble with $\mathcal{A}_X = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$

Example 1 (Uniform Code):

- Let
- Sho

Codes: A Review

Examples

X is an ensemble with $\mathcal{A}_X = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$

Example 1 (Uniform Code):

- Let
- Sho
- All c

4

Codes: A Review

Examples

X is an ensemble with $\mathcal{A}_X = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$

Example 1 (Uniform Code):

- Let
- Sho
- All c
- The *extension* of c maps $\mathtt{aba} \in \mathcal{A}_X^3$

4

Codes: A Review

Examples

X is an ensemble with $\mathcal{A}_X = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$

Example 1 (Uniform Code):

- Let
- Sho
- All c
- The *extension* of c maps $\mathtt{aba} \in \mathcal{A}_X^3$

4

Example 2 (Variable-Length Code):

Codes: A Review

Examples

X is an ensemble with $\mathcal{A}_X = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$

Example 1 (Uniform Code):

- Let
- Sho
- All c
- The *extension* of c maps $\mathtt{aba} \in \mathcal{A}_X^3$

4

Example 2 (Variable-Length Code):

- Let $c(\mathtt{a}) = 0$, $c(\mathtt{b}) = 10$, $c(\mathtt{c}) = 110$, $c(\mathtt{d}) = 111$

Codes: A Review

Examples

X is an ensemble with $\mathcal{A}_X = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$

Example 1 (Uniform Code):

- Let
- Sho
- All c
- The *extension* of c maps $\mathtt{aba} \in \mathcal{A}_X^3$

4

Example 2 (Variable-Length Code):

- Let $c(\mathtt{a}) = 0$, $c(\mathtt{b}) = 10$, $c(\mathtt{c}) = 110$, $c(\mathtt{d}) = 111$
- Shorthand: $C_2 = \{0, 10, 110, 111\}$

Codes: A Review

Examples

X is an ensemble with $\mathcal{A}_X = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$

Example 1 (Uniform Code):

- Let
- Sho
- All c
- The *extension* of c maps $\mathtt{aba} \in \mathcal{A}_X^3$

4

Example 2 (Variable-Length Code):

- Let $c(\mathtt{a}) = 0$, $c(\mathtt{b}) = 10$, $c(\mathtt{c}) = 110$, $c(\mathtt{d}) = 111$
- Shorthand: $C_2 = \{0, 10, 110, 111\}$
- In this case $\ell_1 = 1$, $\ell_2 = 2$, $\ell_3 = \ell_4 = 3$

Codes: A Review

Examples

X is an ensemble with $\mathcal{A}_X = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$

Example 1 (Uniform Code):

- Let
- Sho
- All c
- The *extension* of c maps $\mathtt{aba} \in \mathcal{A}_X^3$ $^+$ 4

Example 2 (Variable-Length Code):

- Let $c(\mathtt{a}) = 0$, $c(\mathtt{b}) = 10$, $c(\mathtt{c}) = 110$, $c(\mathtt{d}) = 111$
- Shorthand: $C_2 = \{0, 10, 110, 111\}$
- In this case $\ell_1 = 1$, $\ell_2 = 2$, $\ell_3 = \ell_4 = 3$
- The *extension* of c maps $\mathtt{aba} \in \mathcal{A}_X^3 \subset \mathcal{A}_X^+$ to 0100

Unique Decodeability

Recall that a code is lossless if for all $x, y \in \mathcal{A}_X$

$$x \neq y \implies c(x) \neq c(y)$$

This ensures that if we work with a single outcome, we can uniquely decode the outcome

When wo
require th

Unique Decodeability

Recall that a code is lossless if for all $x, y \in \mathcal{A}_X$

$$x \neq y \implies c(x) \neq c(y)$$

This ensures that if we work with a single outcome, we can uniquely decode the outcome

When wo
require th

Uniquely Decodable

A code c for X is **uniquely decodable** if no tve the same codeword. That is, for all $\mathbf{x}, \mathbf{y} \in \mathcal{A}_X^+$

$$\mathbf{x} \neq \mathbf{y} \implies c(\mathbf{x}) \neq c(\mathbf{y})$$

This ensures that if we work with a sequence of outcomes, we can still uniquely decode the individual elements

Examples of uniquely decodable codes

Examples:

- $S_1 = \{0001, 0010, 0100, 1000\}$ is uniquely decodable

Examples of uniquely decodable codes

Examples:

- $S_1 = \{0001, 0010, 0100, 1000\}$ is uniquely decodable
  - Uniform + Lossless $\Rightarrow$ Uniquely decodable

Examples of uniquely decodable codes

Examples:

- $C_1 = \{0001, 0010, 0100, 1000\}$ is uniquely decodable
  - Uniform + Lossless $\Rightarrow$ Uniquely decodable
- C_2 = {0, 10, 110, 1110, 1111}

Examples of uniquely decodable codes

Examples:

- $C_1 = \{0001, 0010, 0100, 1000\}$ is uniquely decodable
  - Uniform + Lossless $\Rightarrow$ Uniquely decodable
- C_2 = {0, 10, 110, 111}
  - The code is of course lossless

Examples of uniquely decodable codes

Examples:

- $C_1 = \{0001, 0010, 0100, 1000\}$ is uniquely decodable
  - Uniform + Lossless $\Rightarrow$ Uniquely decodable
- C_2 = {0, 10, 110, 111}
  - The code is of course lossless
  - Lossless $\not\Rightarrow$ Uniquely decodable

Examples of uniquely decodable codes

Examples:

- $C_1 = \{0001, 0010, 0100, 1000\}$ is uniquely decodable
  - Uniform + Lossless $\Rightarrow$ Uniquely decodable
- C_2 = {0
  - The code is of course lossless
  - Lossless $\not\Rightarrow$ Uniquely decodable
- $C_3 = \{0, 10, 110, 111\}$ is uniquely deco

Examples of uniquely decodable codes

Examples:

- $C_1 = \{0001, 0010, 0100, 1000\}$ is uniquely decodable
  - Uniform + Lossless $\Rightarrow$ Uniquely decodable
- C_2 = {0
  - The code is of course lossless
  - Lossless $\not\Rightarrow$ Uniquely decodable
- $C_3 = \{0, 10, 110, 111\}$ is uniquely deco
  - We can easily segment a given code string scanning left to right

Examples of uniquely decodable codes

Examples:

- $C_1 = \{0001, 0010, 0100, 1000\}$ is uniquely decodable
  - Uniform + Lossless $\Rightarrow$ Uniquely decodable
- C_2 = {0
  - The code is of course lossless
  - Lossless $\not\Rightarrow$ Uniquely decodable
- $C_3 = \{0, 10, 110, 111\}$ is uniquely deco
  - We can easily segment a given code string scanning left to right
  - e.g. $0110010 \rightarrow 0, 110, 0, 10$

“Self-punctuating” property

The code $C_3 = \{0, 10, 110, 111\}$ has a “self-punctuating” property

“Self-punctuating” property

The code $C_3 = \{0, 10, 110, 111\}$ has a “self-punctuating” property

Trivial to segment a given code string into individual codewords

- Ke
- On
  stri

“Self-punctuating” property

The code $C_3 = \{0, 10, 110, 111\}$ has a “self-punctuating” property

Trivial to segment a given code string into individual codewords

- Ke
- On
  stri

Once our current segment matches a codeword, n

- Why? No codeword is a prefix of any other

Not true for every uniquely decodable code, e.g. $C_4 = \{0, 01, 011\}$

- First bit 0 $\rightarrow$ no certainty what the symbol is

Prefix Codes

a.k.a *prefix-free* or *instantaneous* codes

A simple property of codes **guarantees** unique decodeability

Prefix property

A codewo + rd
$\mathbf{c}' \in \{0,$ **t**.

Prefix Codes

a.k.a *prefix-free* or *instantaneous* codes

A simple property of codes **guarantees** unique decodeability

Prefix property

A codewo + rd
$\mathbf{c}' \in \{0,$ **t**.

Can you cr

- **Example**: 01101 has prefixes 0, 01, 011, 011

Prefix Codes

a.k.a *prefix-free* or *instantaneous* codes

A simple property of codes **guarantees** unique decodeability

Prefix property

A codewo + rd
$\mathbf{c}' \in \{0,$ **t**.

Can you cr

- **Example**: 01101 has prefixes 0, 01, 011, 011

Prefix Codes

A code $C = \{\mathbf{c}_1, \ldots, \mathbf{c}_I\}$ is a **prefix code** $_i \in C$
there is no prefix of $\mathbf{c}_i$ in C.

In a stream, no confusing one codeword with another

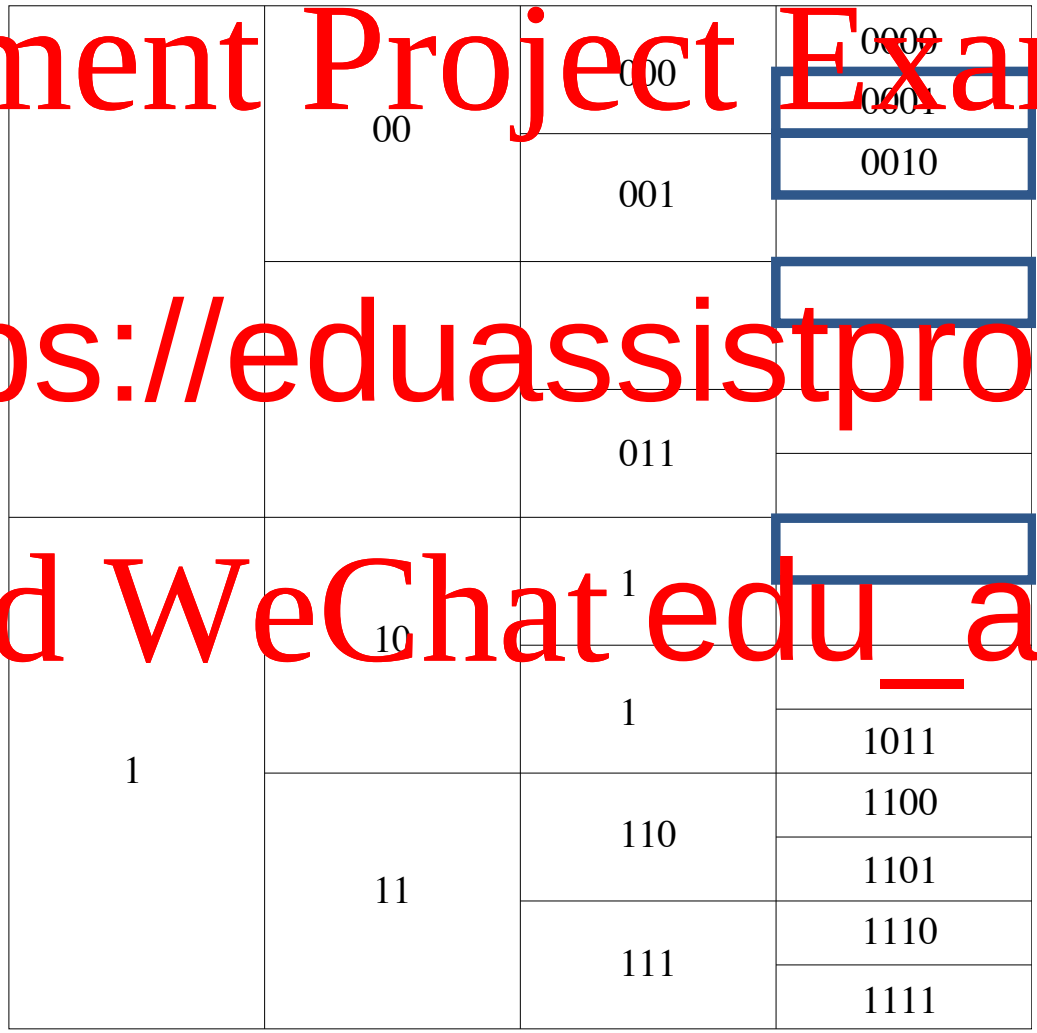
Prefix Codes: Examples

Examples:

- $C_1 = \{0001, 0010, 0100, 1000\}$ is prefix-free
- C_2
- $C_2' = \{1, 10, 110, 111\}$ is *not* prefix fre
- $C_2'' = \{1, 01, 110, 111\}$ is *not* prefix free since $c_3 = 110 = c_1 10$

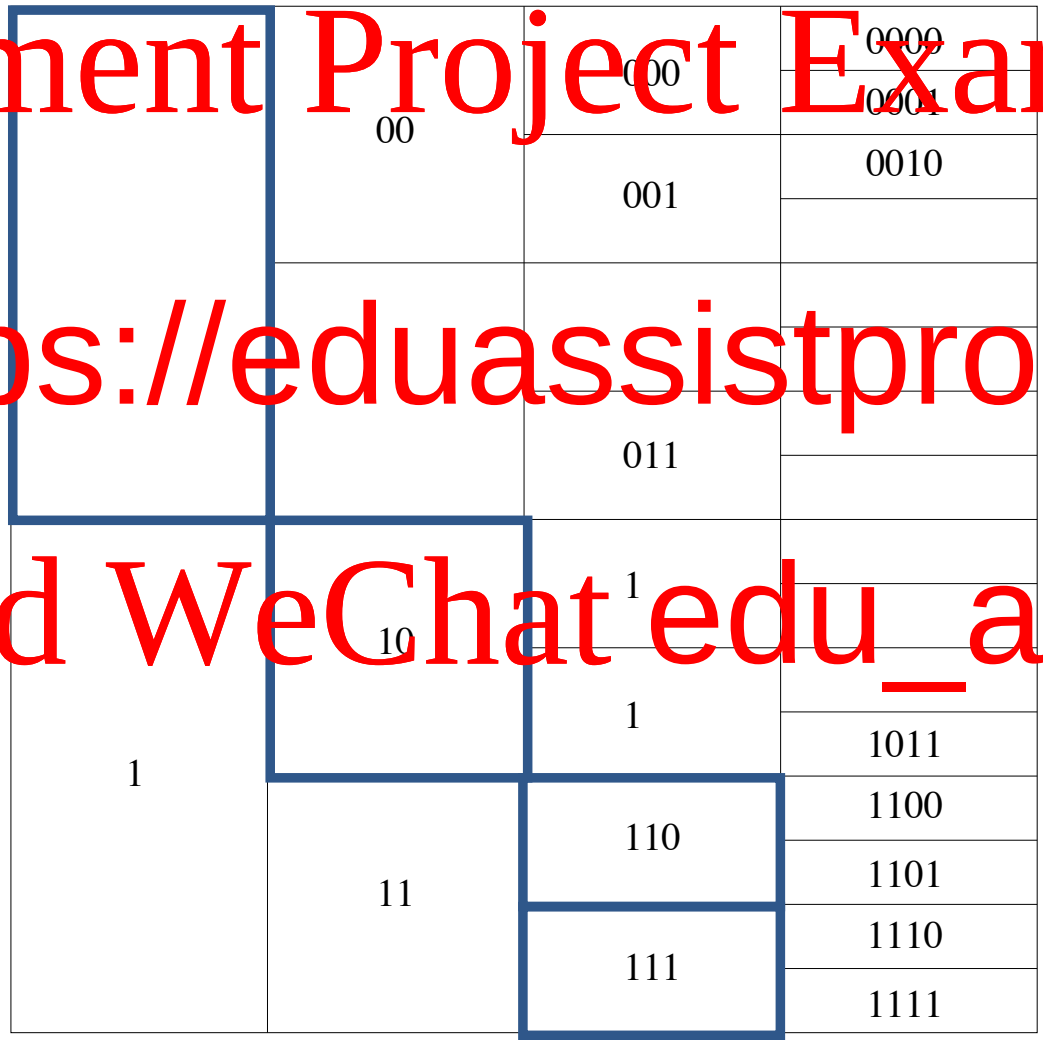
Prefix Codes as Trees

$$C_1 = \{0001, 0010, 0100, 1000\}$$



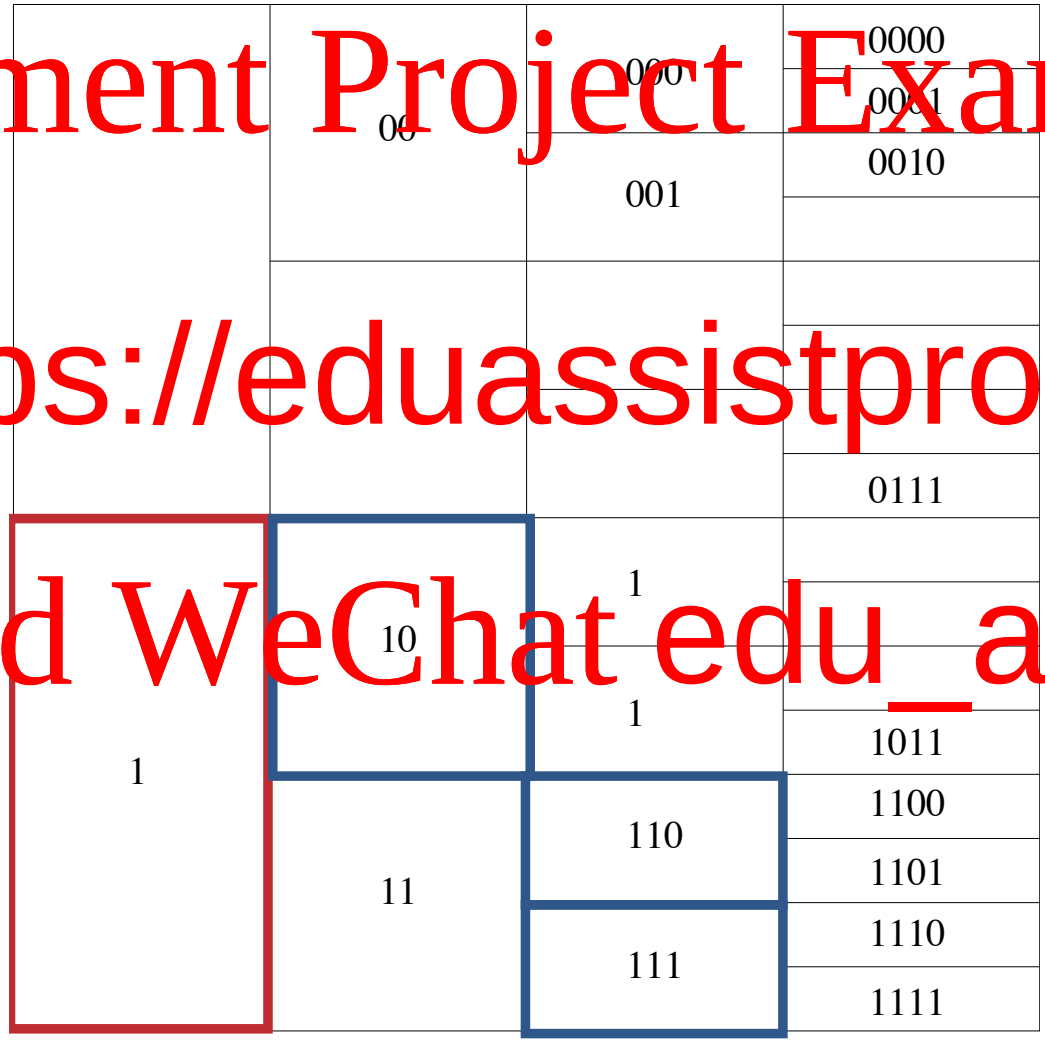
Prefix Codes as Trees

$$C_2 = \{0, 10, 110, 111\}$$



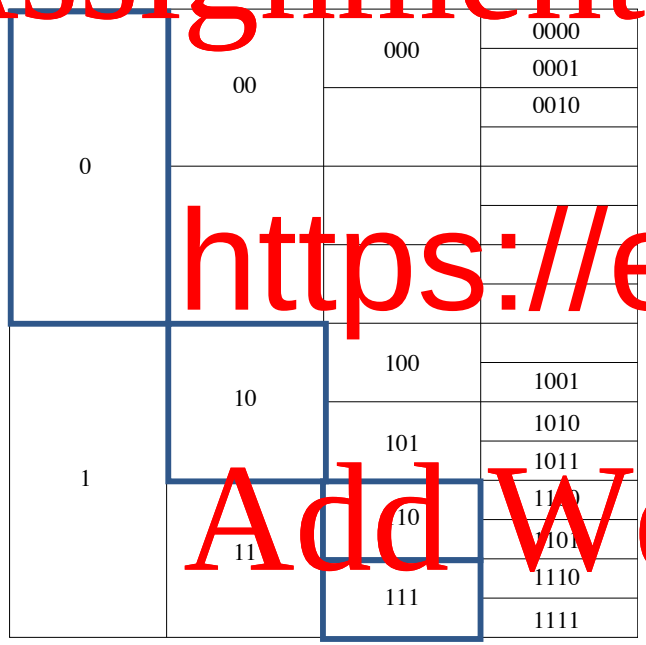
Prefix Codes as Trees

$$C_2' = \{1, 10, 110, 111\}$$



Each codeword choice eliminates its descendants

Prefix Codes are Uniquely Decodeable



- If $\ell^* = \max\{\ell_1, \ldots, \ell_l\}$ then symbol is
  t ℓ^*
- , l l }
  ( ) = ... 1 = a
  - If , d}
  - If
  - If d}

Uniquely Decodeable Codes are Not Always Prefix Codes

A uniquely decodeable code is not necessarily a prefix code

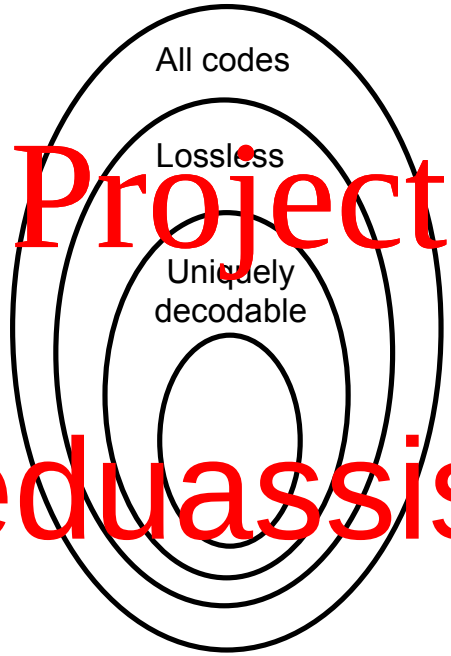
Example: $C_1 = \{0, 01, 011\}$

- 00 . .
- 010
- 011

Example: $C_2 = \{0, 01, 011, 111\}$

- This is the reverse of the prefix code

Relating various types of codes



Note that e.g.

$$\text{Prefix} \implies \text{Uniquely Decodable}$$

but

$$\text{Not Prefix} \not\implies \text{Not Uniquely Decodable}$$

Why prefix codes?

While prefix codes do not represent **all** uniquely decodable codes, they are conve

It will be eas

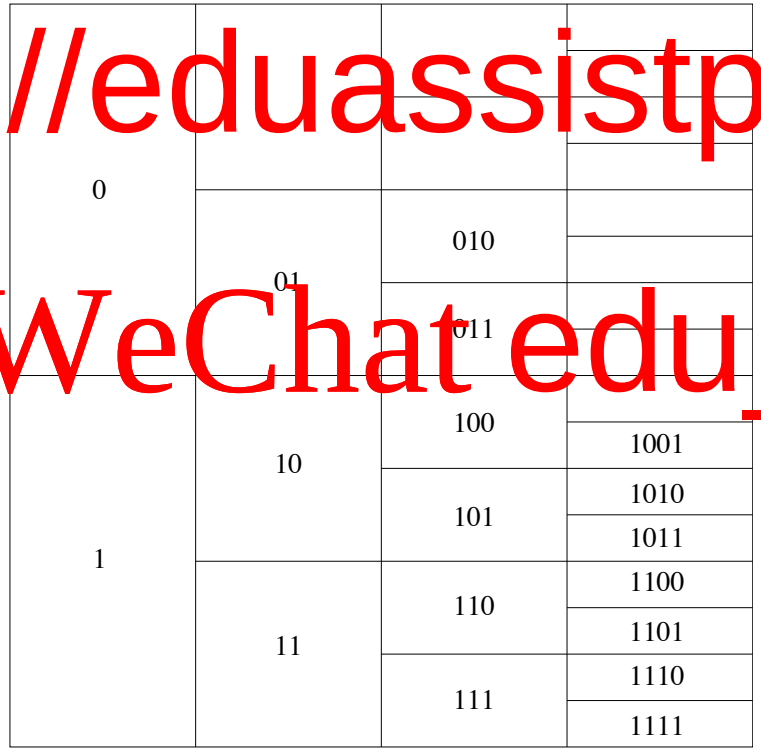
Further, we can quickly establish if a given code is

- Testing for unique decodability is non-trivial i

Lengths and Trees

Suppose someone said “I want prefix codes with codewords lengths”:

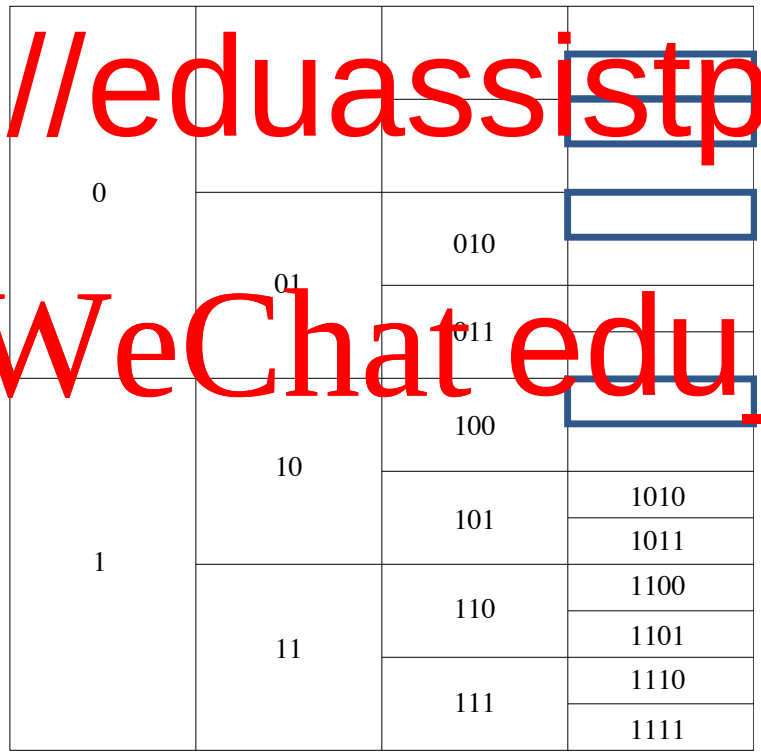
- $L_1 = \{4, 4, 4, 4\}$
- $L_2 = \{1, 2, 3, 3\}$
- $L_3 = \{2, 2, 3, 4, 4\}$
- $L_4 = \{1, 3, 3, 3, 3, 4\}$



Lengths and Trees

Suppose someone said “I want prefix codes with codewords lengths”:

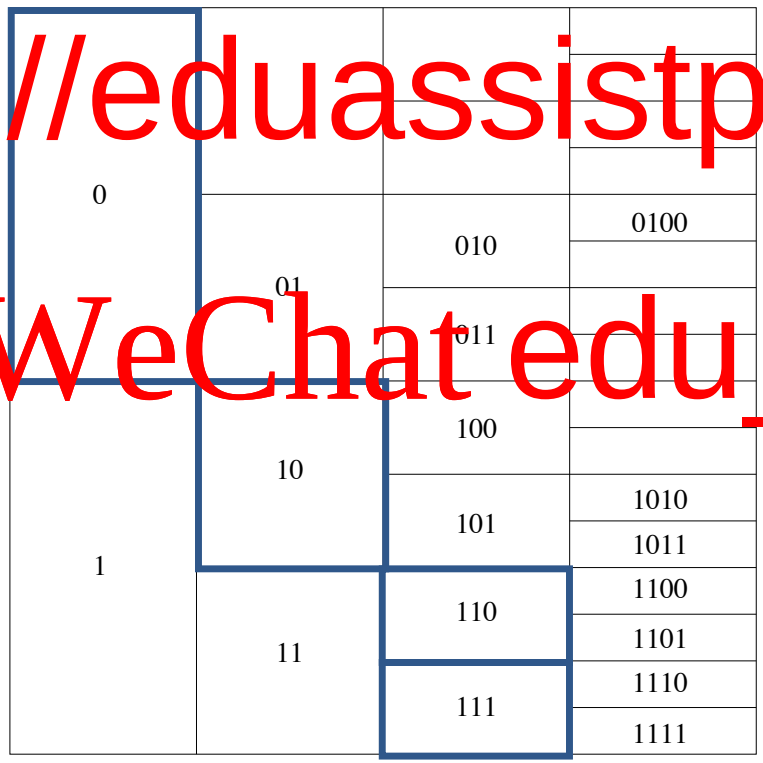
- $L_1 = \{4, 4, 4, 4\}$ — $C_1 = \{0001, 0010, 0100, 1000\}$
- $L_2 = \{1, 2, 3, 3\}$
- $L_3 = \{2, 2, 3, 4, 4\}$
- $L_4 = \{1, 3, 3, 3, 3, 4\}$



Lengths and Trees

Suppose someone said “I want prefix codes with codewords lengths”:

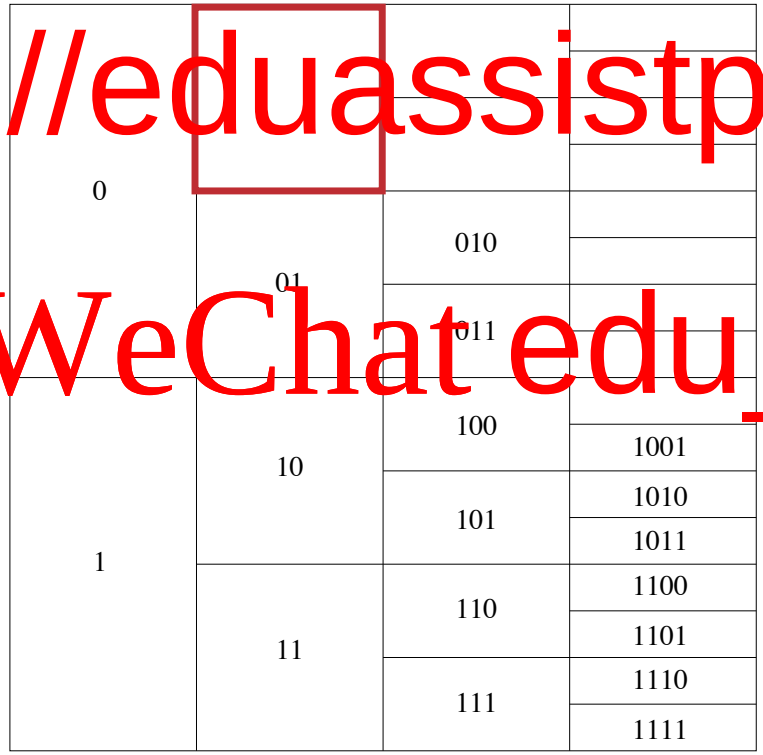
- $L_1 = \{4, 4, 4, 4\}$ — $C_1 = \{0001, 0010, 0100, 1000\}$
- $L_2 = \{1, 2, 3, 3\}$ — $C_2 = \{0, 10, 110, 111\}$
- $L_3 = \{2, 2, 3, 4, 4\}$
- $L_4 = \{1, 3, 3, 3, 3, 4\}$



Lengths and Trees

Suppose someone said “I want prefix codes with codewords lengths”:

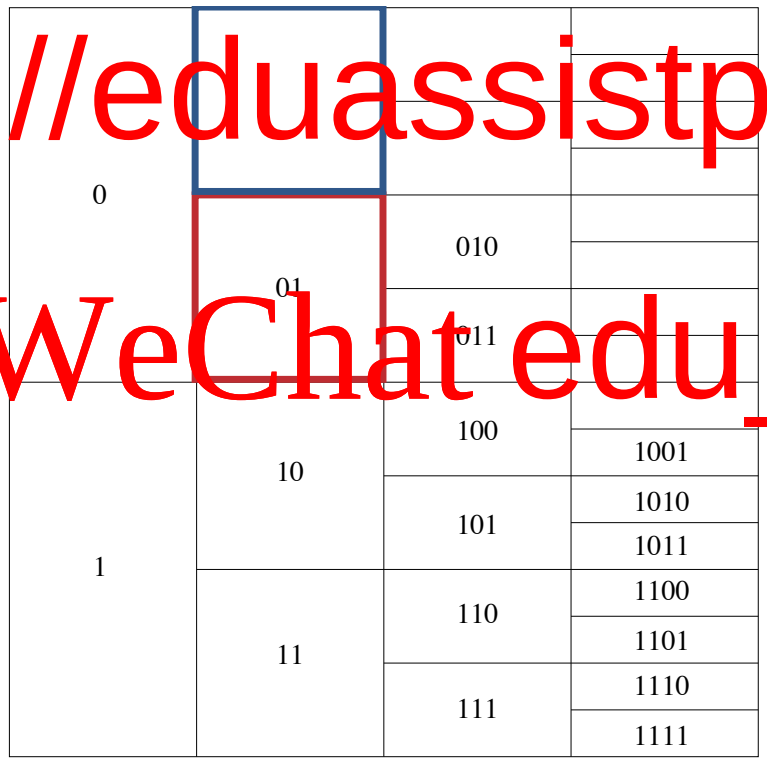
- $L_1 = \{4, 4, 4, 4\}$ — $C_1 = \{0001, 0010, 0100, 1000\}$
- $L_2 = \{1, 2, 3, 3\}$ — $C_2 = \{0, 10, 110, 111\}$
- $L_3 = \{2, 2, 3, 4, 4\}$ — $C_3 = \{00, 01, 100, 1010, 1011\}$
- $L_4 = \{1, 3, 3, 3, 3, 4\}$



Lengths and Trees

Suppose someone said “I want prefix codes with codewords lengths”:

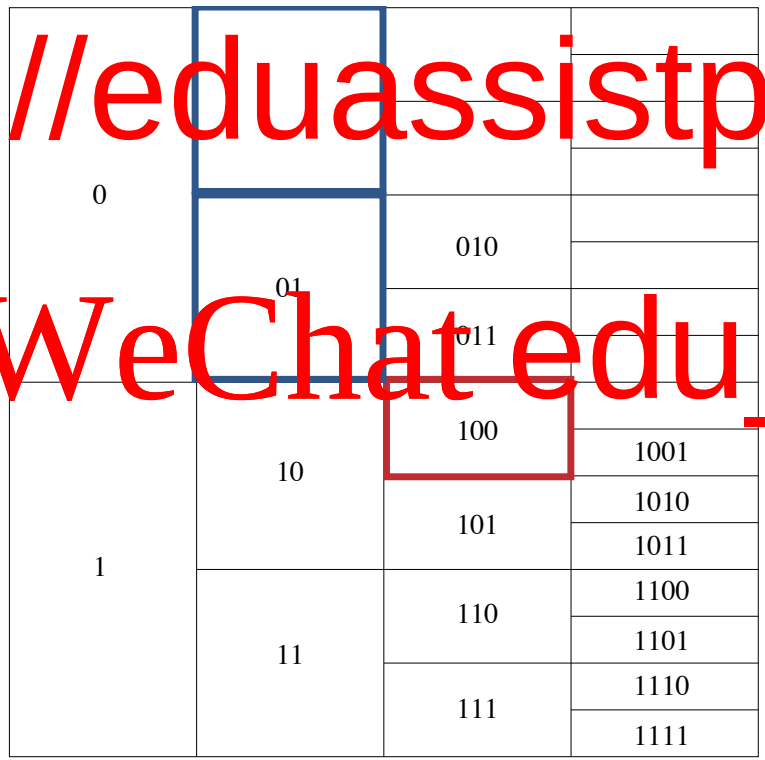
- $L_1 = \{4, 4, 4, 4\}$ — $C_1 = \{0001, 0010, 0100, 1000\}$
- $L_2 = \{1, 2, 3, 3\}$ — $C_2 = \{0, 10, 110, 111\}$
- $L_3 = \{2, 2, 3, 4, 4\}$ — $C_3 = \{00, 01, \ldots\}$
- $L_4 = \{1, 3, 3, 3, 3, 4\}$



Lengths and Trees

Suppose someone said “I want prefix codes with codewords lengths”:

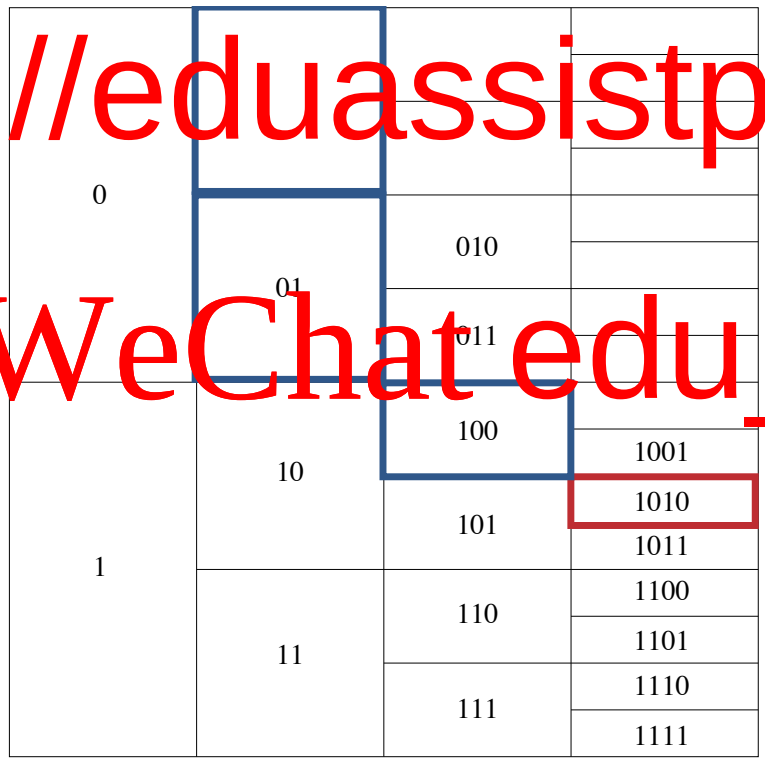
- $L_1 = \{4, 4, 4, 4\}$ — $C_1 = \{0001, 0010, 0100, 1000\}$
- $L_2 = \{1, 2, 3, 3\}$ — $C_2 = \{0, 10, 110, 111\}$
- $L_3 = \{2, 2, 3, 4, 4\}$ — $C_3 = \{00, 01, 100, \ldots\}$
- $L_4 = \{1, 3, 3, 3, 3, 4\}$



Lengths and Trees

Suppose someone said “I want prefix codes with codewords lengths”:

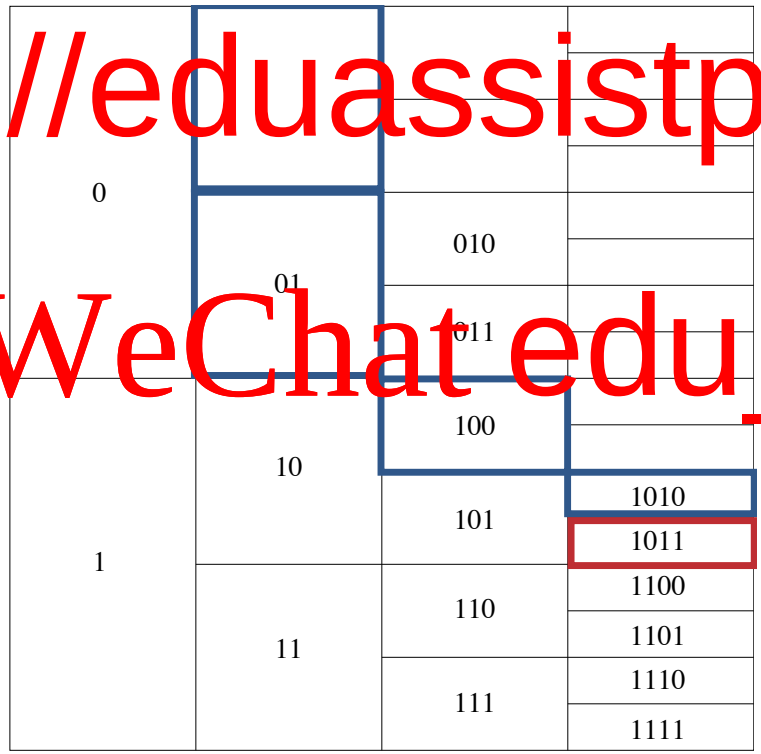
- $L_1 = \{4, 4, 4, 4\}$ — $C_1 = \{0001, 0010, 0100, 1000\}$
- $L_2 = \{1, 2, 3, 3\}$ — $C_2 = \{0, 10, 110, 111\}$
- $L_3 = \{2, 2, 3, 4, 4\}$ — $C_3 = \{00, 01, 100, 1010, \}$
- $L_4 = \{1, 3, 3, 3, 3, 4\}$



Lengths and Trees

Suppose someone said “I want prefix codes with codewords lengths”:

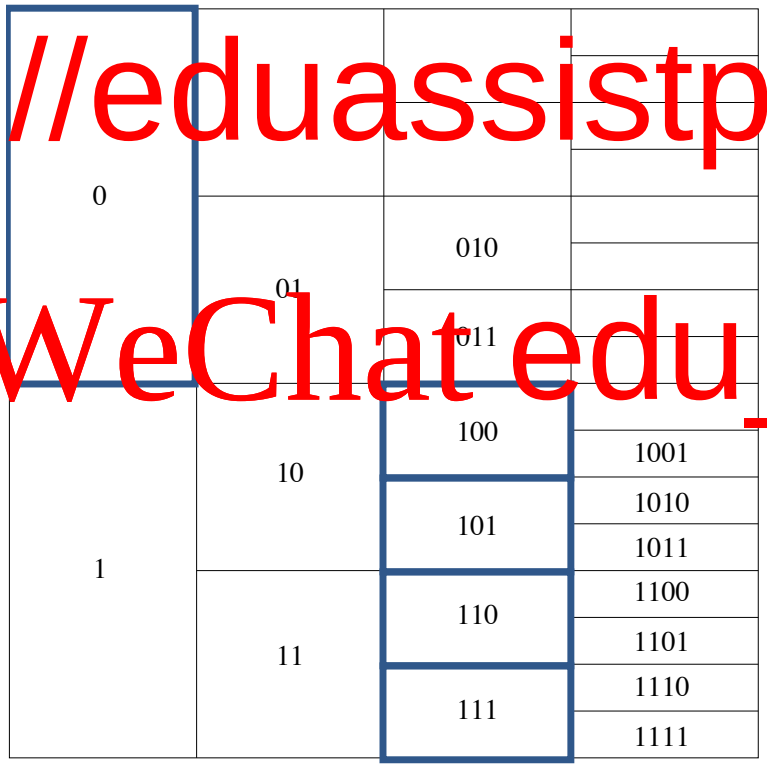
- $L_1 = \{4, 4, 4, 4\}$ — $C_1 = \{0001, 0010, 0100, 1000\}$
- $L_2 = \{1, 2, 3, 3\}$ — $C_2 = \{0, 10, 110, 111\}$
- $L_3 = \{2, 2, 3, 4, 4\}$ — $C_3 = \{00, 01, 100, 1010, 1011\}$
- $L_4 = \{1, 3, 3, 3, 3, 4\}$



Lengths and Trees

Suppose someone said “I want prefix codes with codewords lengths”:

- $L_1 = \{4, 4, 4, 4\}$ — $C_1 = \{0001, 0010, 0100, 1000\}$
- $L_2 = \{1, 2, 3, 3\}$ — $C_2 = \{0, 10, 110, 111\}$
- $L_3 = \{2, 2, 3, 4, 4\}$ — $C_3 = \{00, 01, 100, 1010, 1011\}$
- $L_4 = \{1, 3, 3, 3, 3, 4\}$ — Impossible!



The Kraft Inequality

a.k.a. The Kraft-McMillan Inequality

Kraft Inequality

For any prefix (binary) code C, its codeword lengths $\{\ell_1, \ldots, \ell_I\}$ satisfy

$$\sum_{i=1}^{I} \frac{1}{2^{\ell_i}} \leq 1$$

Convers $\{\ell_1, \ldots, \ell_I\}$ de
C with those codeword lengths.

The Kraft Inequality

a.k.a. The Kraft-McMillan Inequality

Kraft Inequality

For any prefix (binary) code C, its codeword lengths $\{\ell_1, \ldots, \ell_I\}$ satisfy

$$\sum_{i=1}^{I} 2^{-\ell_i} \leq 1$$

Convers $\{\ell_1, \ldots, \ell_I\}$ de
C with those codeword lengths.

Examples:

1. $C_1 = \{0001, 0010, 0100, 1000\}$ is prefix and $\sum_{i=1}^{4} 2^{-4} = \frac{1}{4} \leq 1$

The Kraft Inequality

a.k.a. The Kraft-McMillan Inequality

Kraft Inequality

For any prefix (binary) code C, its codeword lengths $\{\ell_1, \ldots, \ell_I\}$ satisfy

$$\sum_{i=1}^{I} 2^{-\ell_i} \leq 1$$

Conversely, if $\{\ell_1, \ldots, \ell_I\}$ satisfy the above inequality, then there exists a prefix code C with those codeword lengths.

Examples:

1. $C_1 = \{0001, 0010, 0100, 1000\}$ is prefix and $\sum_{i=1}^{4} 2^{-4} = \frac{1}{4} \leq 1$
2. $C_2 = \{0, 10, 110, 111\}$ is prefix and $\sum_{i=1}^{4} 2^{-\ell_i} = \frac{1}{2} + \frac{1}{4} + \frac{2}{8} = 1$

The Kraft Inequality

a.k.a. The Kraft-McMillan Inequality

Kraft Inequality

For any prefix (binary) code C, its codeword lengths $\{\ell_1, \ldots, \ell_I\}$ satisfy

$$\sum_{i=1}^{I} 2^{-\ell_i} \leq 1$$

Convers $\{\ell_1, \ldots, \ell_I\}$ de
C with those codeword lengths.

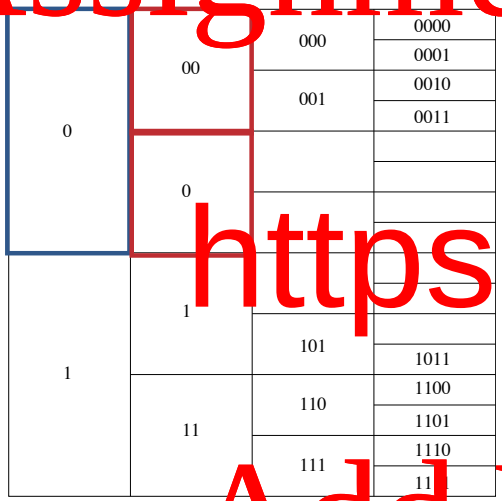
Examples:

1. $C_1 = \{0001, 0010, 0100, 1000\}$ is prefix and $\sum_{i=1}^{4} 2^{-4} = \frac{1}{4} \leq 1$
2. $C_2 = \{0, 10, 110, 111\}$ is prefix and $\sum_{i=1}^{4} 2^{-\ell_i} = \frac{1}{2} + \frac{1}{4} + \frac{2}{8} = 1$
3. Lengths $\{1, 2, 2, 3\}$ give $\sum_{i=1}^{4} 2^{-\ell_i} = \frac{1}{2} + \frac{2}{4} + \frac{1}{8} > 1$ so no prefix code

Prefixes Exclude Codes

We are constrained when constructing prefix codes, as selecting a codeword eliminates a whole subtree

Choosing a prefix codeword of length 1 — e.g., $c(\mathtt{a}) = 0$ — excludes:

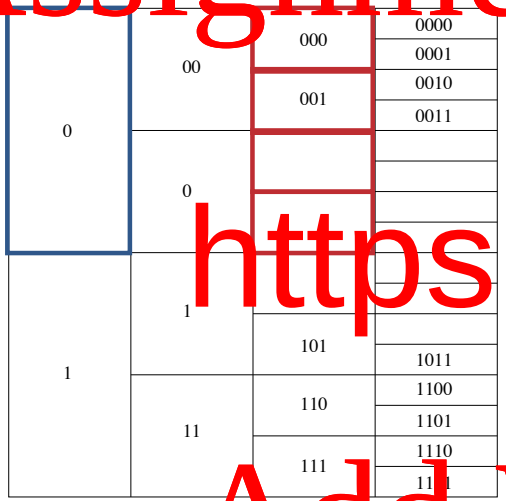


- 2 x 2-bit codewords: 00, 01

Prefixes Exclude Codes

We are constrained when constructing prefix codes, as selecting a codeword eliminates a whole subtree

Choosing a prefix codeword of length 1 — e.g., $c(a) = 0$ — excludes:

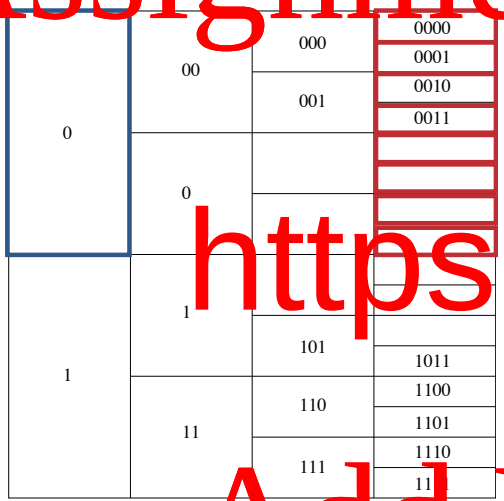


- 2 x 2-bit codewords: 00, 01
- , 011}

Prefixes Exclude Codes

We are constrained when constructing prefix codes, as selecting a codeword eliminates a whole subtree

Choosing a prefix codeword of length 1 — e.g., $c(a) = 0$ — excludes:



- 2 x 2-bit codewords: $00, 01$
- $, 011\}$
- $01 11\}$

Prefixes Exclude Codes

We are constrained when constructing prefix codes, as selecting a codeword eliminates a whole subtree

Choosing a prefix codeword of length 1 — e.g., $c(a) = 0$ — excludes:

- 2 x 2-bit codewords: 00, 01
- , 011}
- 011}
- $2^{k-\ell}$ x k-bit co

Prefixes Exclude Codes

We are constrained when constructing prefix codes, as selecting a codeword eliminates a whole subtree

Choosing a prefix codeword of length 1 — e.g., $c(a) = 0$ — excludes:

- 2 x 2-bit codewords: 00, 01
- , 011}
- 011}
- $2^{k-\ell}$ x k-bit co

Prefixes Exclude Codes

We are constrained when constructing prefix codes, as selecting a codeword eliminates a whole subtree

Choosing a prefix codeword of length 1 — e.g., $c(a) = 0$ — excludes:

- 2 x 2-bit codewords: 00, 01
- , 011}
- 011}
- $2^{k-\ell}$ x k-bit co

For lengths $L = \{\ell_1, \ldots, \ell_I\}$ and $\ell^* = \max\{\ell_1, \ldots$

$$\sum_{i=1}^{I} 2^{\ell^* - \ell_i}$$

excluded ℓ^*-bit codewords.

Prefixes Exclude Codes

We are constrained when constructing prefix codes, as selecting a codeword eliminates a whole subtree

Choosing a prefix codeword of length 1 — e.g., $c(a) = 0$ — excludes:

- 2 x 2-bit codewords: 00, 01
- , 011}
- 011}
- $2^{k-\ell}$ x k-bit co

For lengths $L = \{\ell_1, \ldots, \ell_I\}$ and $\ell^* = \max\{\ell_1, \ldots$

$$\sum_{i=1}^{I} 2^{\ell^* - \ell_i} \leq 2^{\ell^*}$$

excluded ℓ^*-bit codewords. But there are only 2^{ℓ^*} possible ℓ^*-bit codewords

Prefixes Exclude Codes

We are constrained when constructing prefix codes, as selecting a codeword eliminates a whole subtree

Choosing a prefix codeword of length 1 — e.g., $c(a) = 0$ — excludes:

- 2 x 2-bit codewords: 00, 01
- , 011}
- 011}
- $2^{k-\ell}$ x k-bit co

For lengths $L = \{\ell_1, \ldots, \ell_I\}$ and $\ell^* = \max\{\ell_1, \ldots$

$$\frac{1}{2^{\ell^*}} \sum_{i=1}^{I} 2^{\ell^* - \ell_i} \leq 1$$

excluded ℓ^*-bit codewords. But there are only 2^{ℓ^*} possible ℓ^*-bit codewords

Prefixes Exclude Codes

We are constrained when constructing prefix codes, as selecting a codeword eliminates a whole subtree

Choosing a prefix codeword of length 1 — e.g., $c(a) = 0$ — excludes:

- 2 x 2-bit codewords: 00, 01
- , 011}
- 011}
- $2^{k-\ell}$ x k-bit co

For lengths $L = \{\ell_1, \ldots, \ell_I\}$ and $\ell^* = \max\{\ell_1, \ldots$

$$\sum_{i=1}^{I} 2^{-\ell_i} \leq 1$$

excluded ℓ^*-bit codewords. But there are only 2^{ℓ^*} possible ℓ^*-bit codewords

Kraft inequality: other direction

Suppose we are given lengths satisfying

$$\sum_{i=1}^{I} 2^{-l_i} \leq 1$$

Then, we c

- Pic first cod
- Removing all descendants of the node (to ens
- Picking the next (remaining) node at depth second codeword
- Removing all descendants of the node (to ensure the prefix condition)
- ⋮

Kraft inequality: comments

Kraft's inequality actually holds more generally for uniquely decodable codes

- Harder to prove

Note that if

$$\sum_{i=1}^{I} 2^{-\ell_i} \leq$$

it does not mean the **given** code necessaril

- Just that we can **construct** a prefix code with these lengths

Summary

Key ideas from this lecture:

- **Prefix** and **Uniquely Decodeable** variable-length codes
- Prefix codes are tree-like
- Eve
- The
  - $_i$ $^i \leq$ xists
  - Prefix/U.D. code implies $\sum_i 2^{-\ell_i} \leq$

Relevant Reading Material:

- MacKay: §5.1 and §5.2
- Cover & Thomas: §5.1, §5.2, and §5.5

Next time

Bound on expected length for a prefix code

Shannon codes

Huffman coding