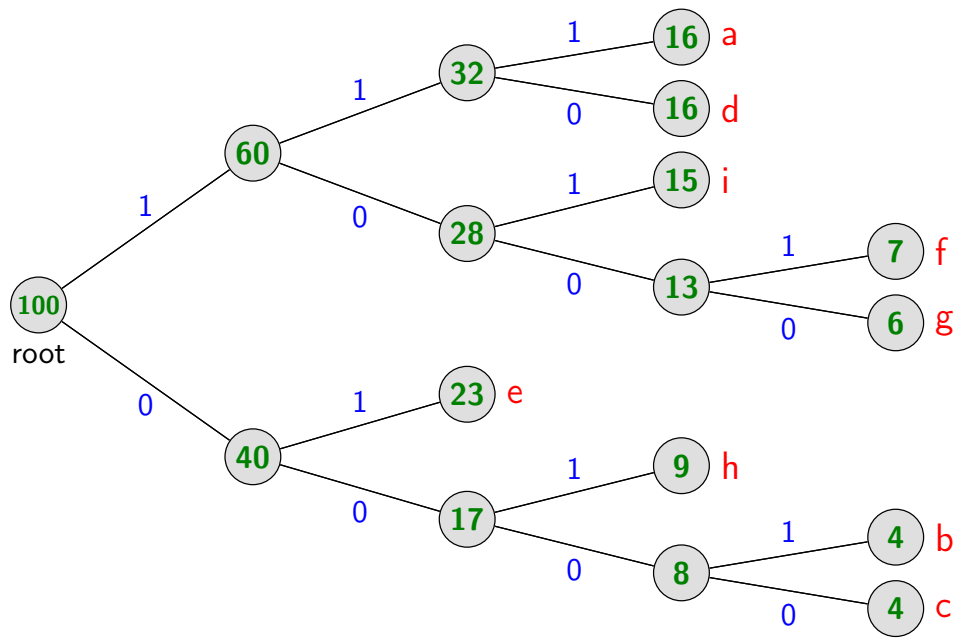
Lossless Coding II

a_k	p_k	$\boldsymbol{b}_k$
a	0.16	111
b	0.04	0001
c	0.04	0000
d	0.16	110
e	0.23	01
f	0.07	1001
g	0.06	1000
h	0.09	001
i	0.15	101



Variable-Length Codes for Scalars

Scalar Variable-Length Codes

- Consider individual symbols s_n of message $\boldsymbol{s}^{(L)} = \{s_0, s_1, \cdots, s_{L-1}\}$
- Assign a codeword $\boldsymbol{b}_n^{(\ell_n)}$ with ℓ_n bits (or length ℓ_n) to each symbol s_n

$$\boldsymbol{b}_n^{(\ell_n)} = \gamma(s_n)\,, \qquad \ell_n = \dim(\,\gamma(s_n)\,) \tag{1}$$

- Form bitstream $\boldsymbol{b}^{(L)}$ by concatenating codewords $\left\{\boldsymbol{b}_n^{(\ell_n)}\right\}$
- Source code γ represents a table that maps symbols to codewords

Example

alphabet

$\mathcal{A} = \{\mathrm{A}, \mathrm{B}, \mathrm{M}, \mathrm{N}\}$

lossless code γ

letter	codeword
A	0
B	110
M	111
N	10

$\boldsymbol{s} =$ "BANANAMAN"

$\Updownarrow$

$\boldsymbol{b} =$ "110010010011101 0"

Efficiency of Scalar Variable-Length Codes

Assumptions

- Messages $\boldsymbol{s}^{(L)}$ represent finite-length realizations (with length L) of a **stationary discrete random process** $\boldsymbol{S} = \{S_0, S_1, \cdots\}$
- Random variables $S_n = S$ have a countable alphabet $\mathcal{A} = \{a_0, a_1, a_2, \cdots\}$
- Marginal pmf $p_S(a)$ for the random variables S is known

$$p_k = p_S(a_k) = \mathrm{P}(S = a_k) \qquad \forall a_k \in \mathcal{A}$$

Characterizing the Efficiency

- Codeword lengths ℓ_n: Function of the random variables S_n

$$\ell_n = \ell(S_n) = \dim(\,\gamma(S_n)\,) \tag{2}$$

- Efficiency measure: **Average codeword length $\bar{\ell}$ per symbol**

$$\bar{\ell} = \mathrm{E}\{\,\ell(S)\,\} = \sum_{\forall a_k \in \mathcal{A}} \ell(a_k)\, p_S(a_k) = \sum_k \ell_k\, p_k \tag{3}$$

Construction of Lossless Codes

Design Goals

1. Minimize average codeword length $\bar{\ell}$

$$\bar{\ell} = \mathrm{E}\{\, \ell(S) \,\} = \sum_k \ell_k \, p_k$$

2. Retain **unique decodability** of arbitrarily long messages !

Code Examples

a_k	p_k	code A	code B	code C	code D	code E
a	0.5	0	0	0	00	0
b	0.25	10	01	01	01	10
c	0.125	11	010	011	10	110
d	0.125	11	011	111	110	111
	$\bar{\ell}$	1.5	1.75	1.75	2.125	1.75
uniquely decodable?		**no** (***singular***)	**no** (c=b,a)	**yes** (*delay*)	**yes** (***instantaneous codes***)	**yes**

Unique Decodability

Uniquely Decodable Codes

- Necessary condition: Non-singular codes

$$\forall a, b \in \mathcal{A} : a \neq b, \qquad \gamma(a) \neq \gamma(b) \tag{4}$$

➡ Not sufficient (see code B in example)

➡ Require: **Each sequence of bits can only be generated by one possible sequence of source symbols**

Prefix Codes

- One class of uniquely decodable codes
- Property: No codeword for an alphabet letter represents the codeword or a prefix of the codeword for any other alphabet letter

➡ Obvious: Any concatenation of codewords can be uniquely decoded

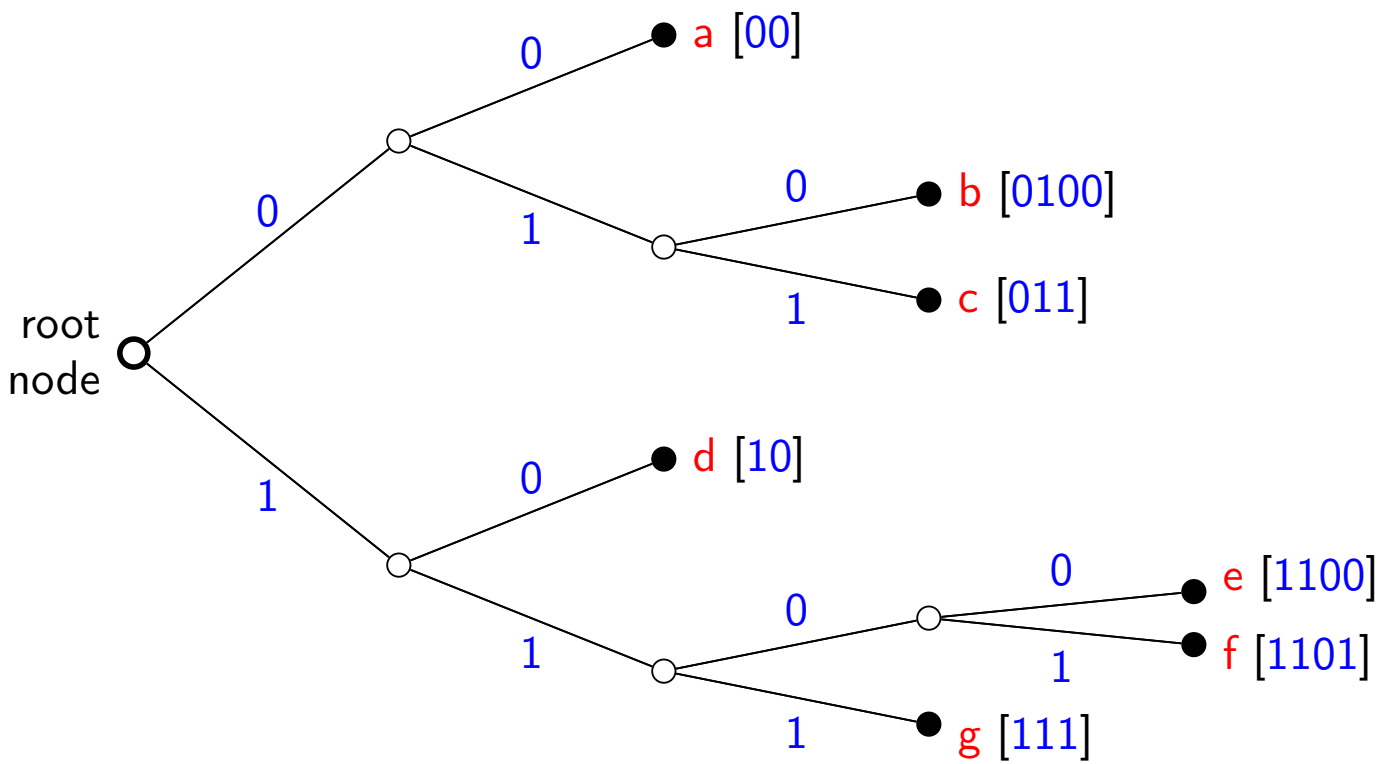
- Also referred to as **prefix-free codes** or **instantaneous codes**

Binary Code Trees for Prefix Codes

Prefix codes can be represented as binary code trees

- Alphabet letters are assigned to terminal nodes
- The two branches starting from a node are labeled with "0" and "1"
- Codewords are given by labels on path from the root to a terminal node

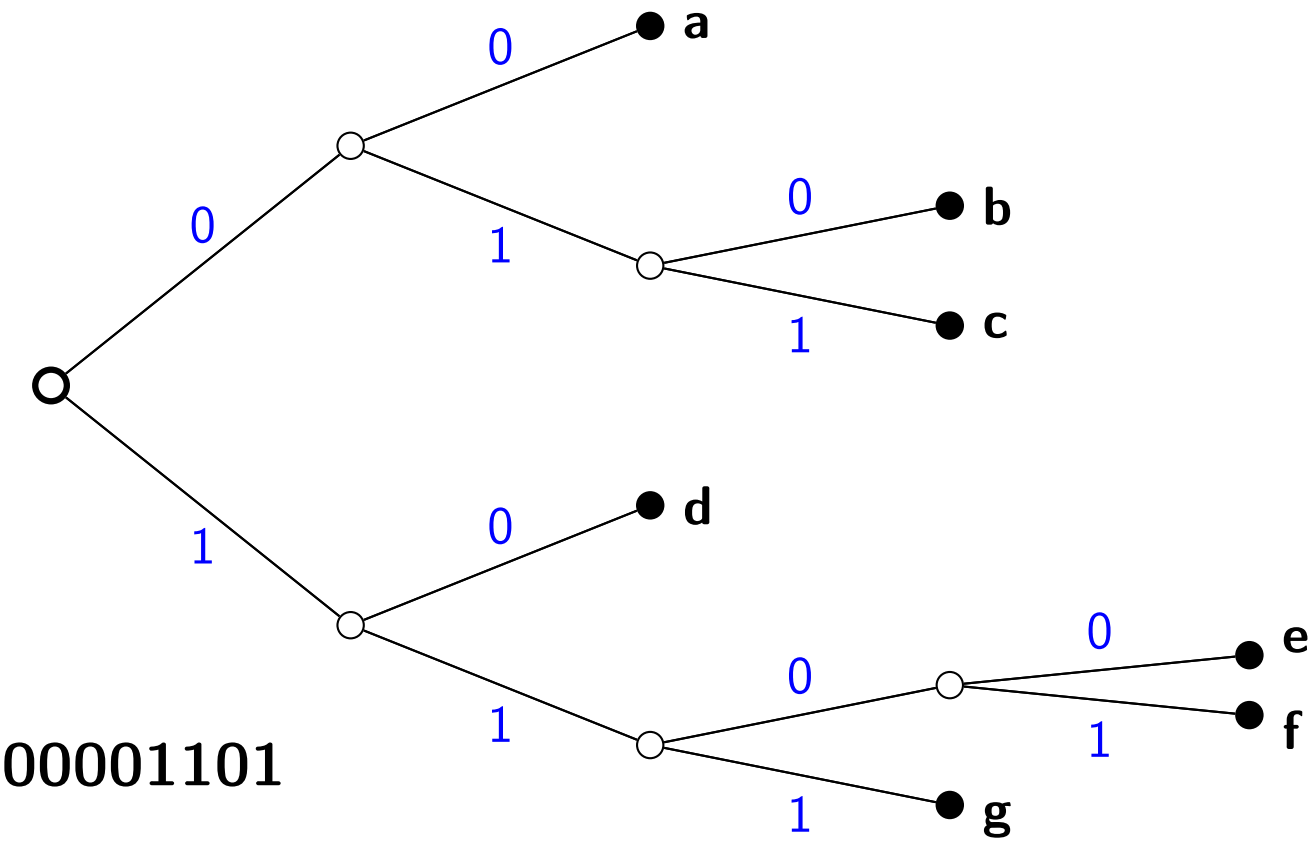
letter	codeword
a	00
b	010
c	011
d	10
e	1100
f	1101
g	111



Example: Parsing for Prefix Codes

➡ Read bit by bit and follow code tree from root to terminal node

letter	codeword
a	00
b	010
c	011
d	10
e	1100
f	1101
g	111



bitstream: **0101100001101**

symbols: **beaf**

Encoding and Decoding for Prefix Codes

Remember: Encoding

- Concatenate codewords for individual symbols of a message

➡ Valid for all scalar variable length codes

Decoding Algorithm using Binary Tree Representation

1. Set the current node n_{curr} equal to the root node
2. Read the next bit b from the bitstream and follow the branch labeled with b from the current node n_{curr} to the corresponding child node n_{child}
3. Check whether the reached child node n_{child} is a terminal node
   - If yes: Return the associated alphabet letter and proceed with 1
   - If no: Move the current node, $n_{\mathrm{curr}} = n_{\mathrm{child}}$, and proceed with 2

Important Property of Prefix Codes

- Not only **uniquely decodable**, but also **instantaneously decodable**

➡ Can output each symbol as soon as the last bit of its codeword is read

Advantage of Instantaneous Decodability

...	
coding_mode	code B
if(coding_mode == 2) {	
intra_prediction_mode	code C
} else if(coding_mode == 1) {	
motion_vector_abs_x	code D
if(motion_vector_abs_x > 0)	
motion_vector_sign_x	code A
motion_vector_abs_y	code D
if(motion_vector_abs_y > 0)	
motion_vector_sign_y	code A
}	
residual_data_flag	code A
...	

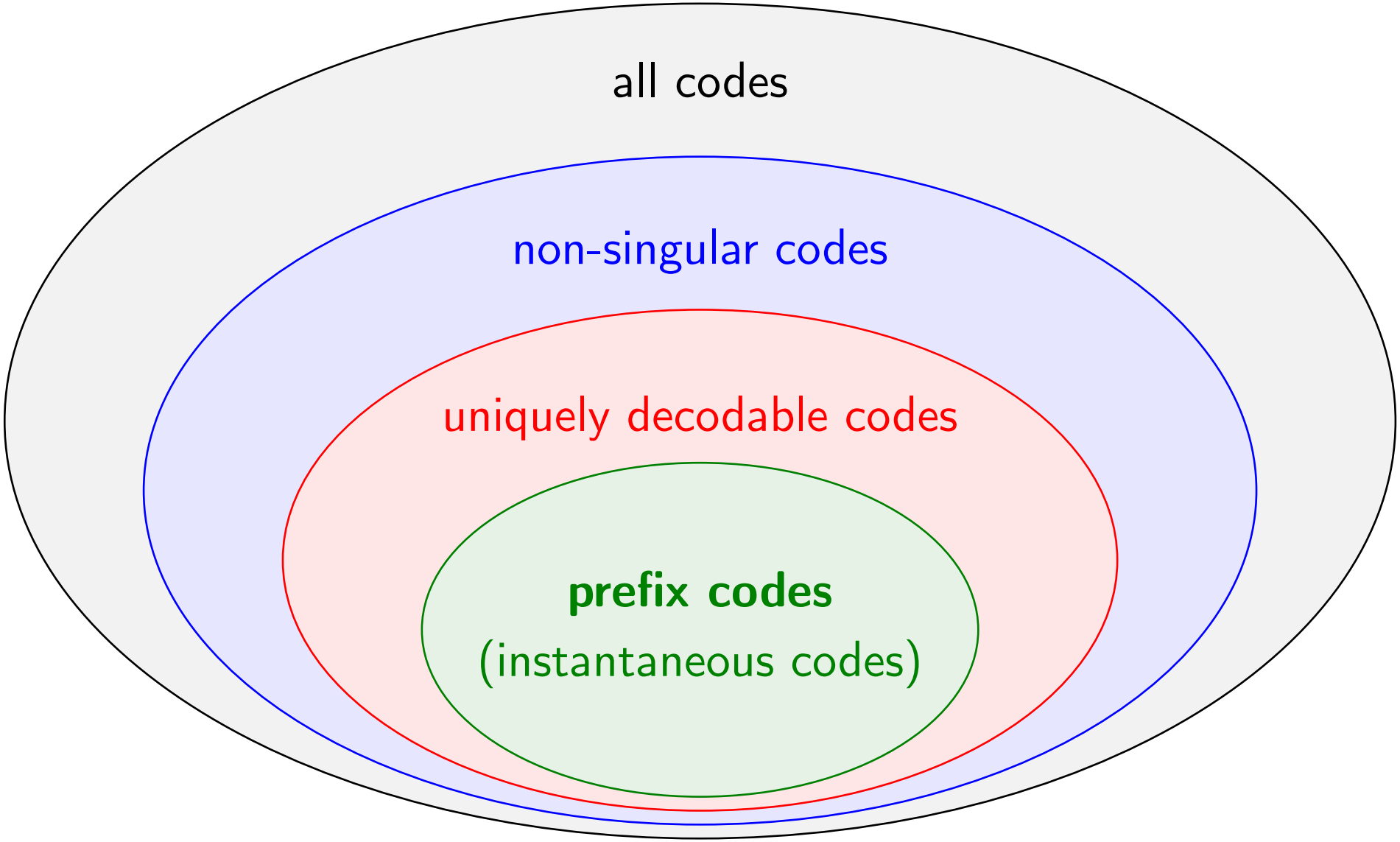
symbol index k	code A	code B	code C	code D
0	0	0	00	1
1	1	10	010	01
2		11	1	001
3			011	0001
4				00001
5				000001
6				0000001
7				00000001
8				000000001
9				0000000001
⋮				⋮

Many applications: Mixture of different types of symbols (different alphabets)

- Order and presence of symbols is given by a **syntax**
- Resulting bitstream must be uniquely decodable

➡ Straightforward with prefix codes:
Switch code after a symbol is decoded (due to instantaneous decodability)

Classification of Codes



Intermediate Results / Open Questions

Prefix Codes

- Uniquely decodable codes
- Can be represented by binary code tree

→ Simple encoding and decoding algorithms

→ Instantaneously decodable

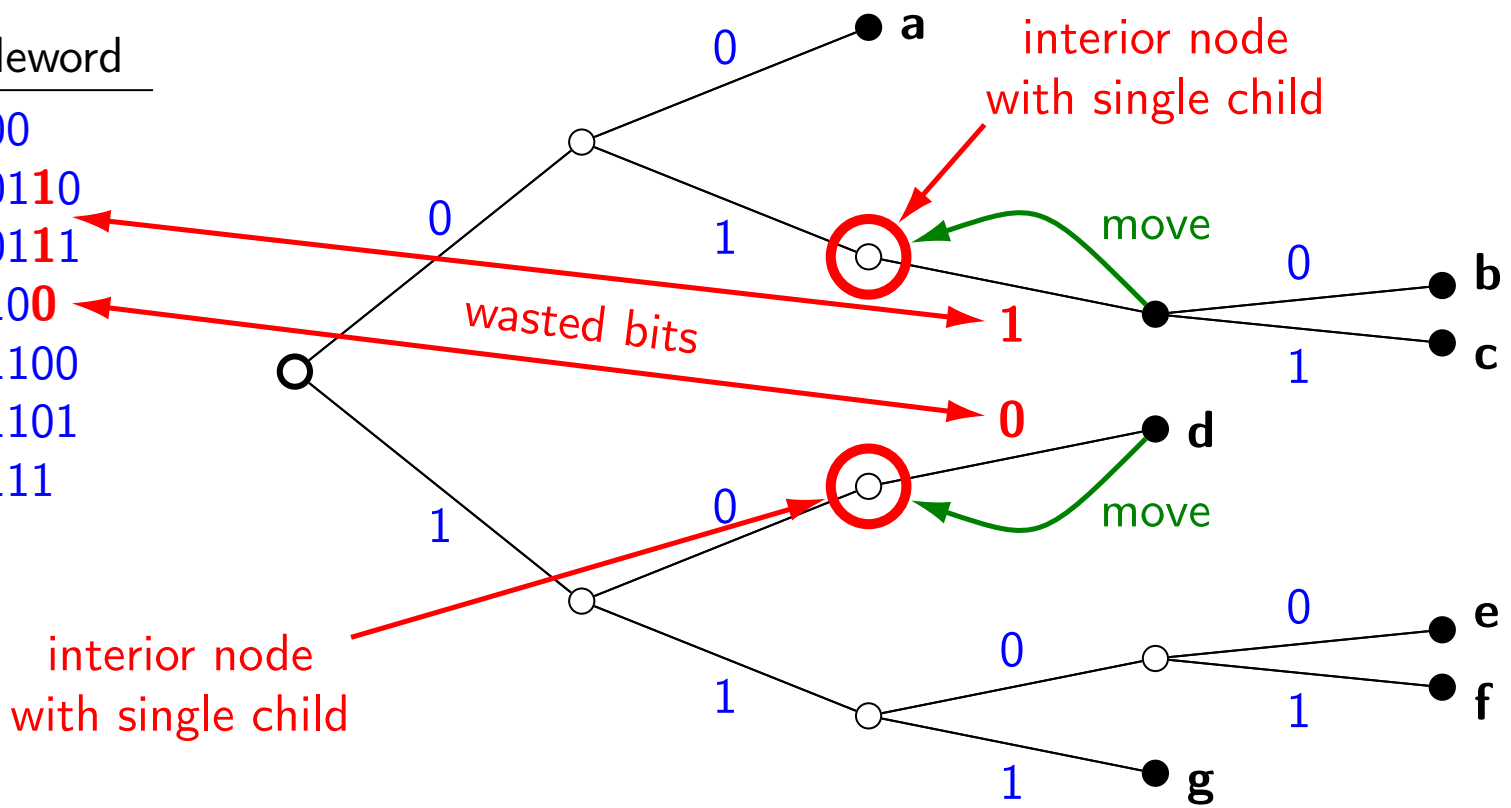
→ Straightforward use in complicated syntax

Open Questions

1. Are there any other uniquely decodable codes that can achieve a smaller average codeword length than the best prefix code?
2. What is the **minimum average codeword length** that can be achieved for a given source?
3. How can we develop an **optimal code** for a source with given pmf?

Prefix Codes with Structural Redundancy

letter	codeword
a	00
b	0110
c	0111
d	100
e	1100
f	1101
g	111



Binary code tree is **not a full binary tree** (also: improper binary tree)

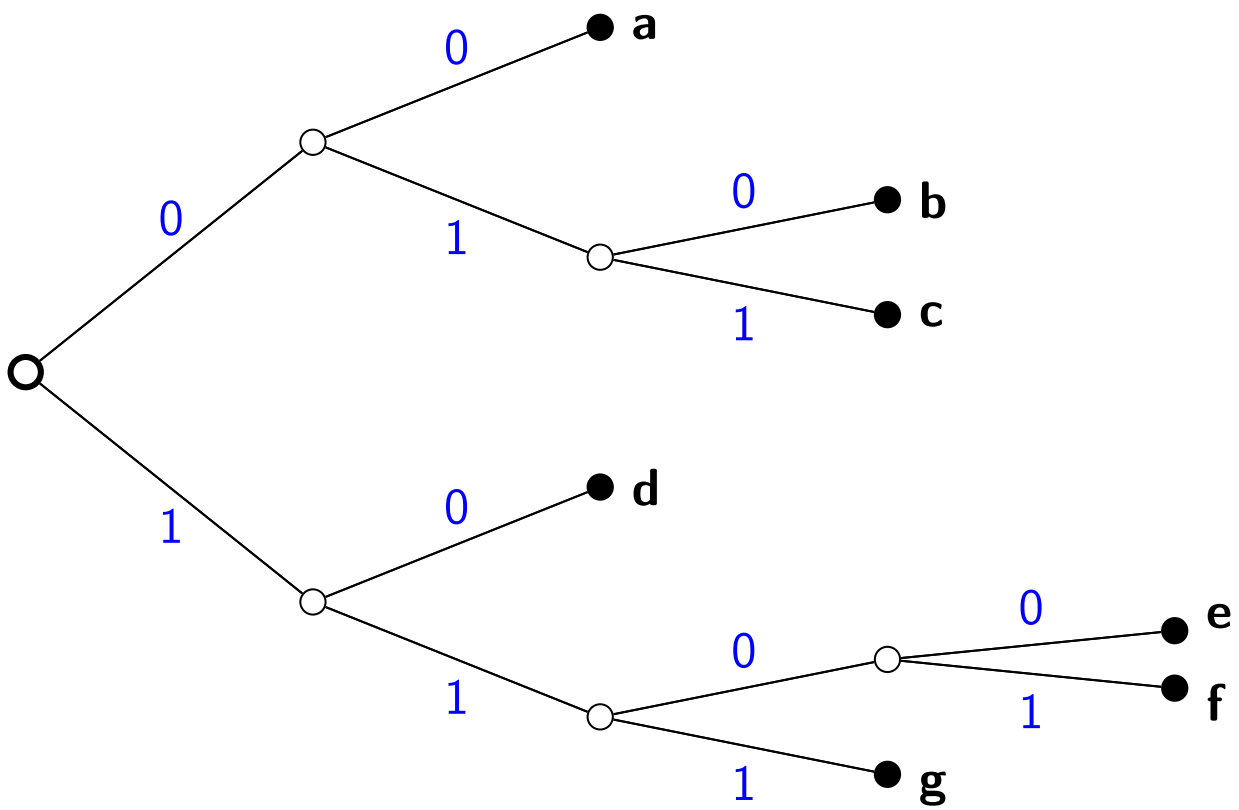
- There are interior nodes with only one child

➡ Results in wasted bit (for one or more codewords)

➡ Average codeword length can be decreased by **moving single child node(s)**

Prefix Codes without Structural Redundancy

letter	codeword
a	00
b	010
c	011
d	10
e	1100
f	1101
g	111



Binary code tree is a **full binary tree** (also: proper binary tree)

- All nodes have either no or two childs

➡ All bits in codewords are required

➡ But: The code may still be inefficient for a given source

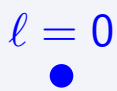
Measure for Structural Redundancy of Prefix Codes

Measure for Structural Redundancy

- Consider measure

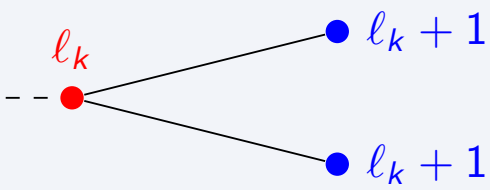
$$\zeta(\gamma) = \sum_{\forall k} 2^{-\ell_k} \tag{5}$$

- Root node



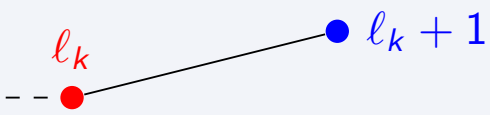
$$\zeta(\gamma)_{\text{root}} = 2^0 = 1 \tag{6}$$

- Adding two childs at node with ℓ_k



$$\begin{aligned} \zeta(\gamma)_{\text{new}} &= \zeta(\gamma)_{\text{old}} - 2^{-\ell_k} + 2 \cdot 2^{-(\ell_k+1)} \\ &= \zeta(\gamma)_{\text{old}} \end{aligned} \tag{7}$$

- Adding one child at node with ℓ_k



$$\begin{aligned} \zeta(\gamma)_{\text{new}} &= \zeta(\gamma)_{\text{old}} - 2^{-\ell_k} + 2^{-(\ell_k+1)} \\ &< \zeta(\gamma)_{\text{old}} \end{aligned} \tag{8}$$

Measure for Structural Redundancy of Prefix Codes

We have shown ...

- Prefix codes without structural redundancy (full binary code tree)

$$\zeta(\gamma) = \sum_{\forall k} 2^{-\ell_k} = 1 \tag{9}$$

- Prefix codes with structural redundancy (not a full binary code tree)

$$\zeta(\gamma) = \sum_{\forall k} 2^{-\ell_k} < 1 \tag{10}$$

- Prefix codes always have

$$\zeta(\gamma) = \sum_{\forall k} 2^{-\ell_k} \leq 1 \tag{11}$$

Construction Of Prefix Codes For Given Codeword Lengths

Given

- Ordered set of codeword lengths $\{\ell_0, \ell_1, \ell_2, \cdots\}$, with $\ell_0 \leq \ell_1 \leq \ell_2, \leq \cdots$, that satisfies

$$\zeta(\gamma) = \sum_{\forall k} 2^{-\ell_k} \leq 1$$

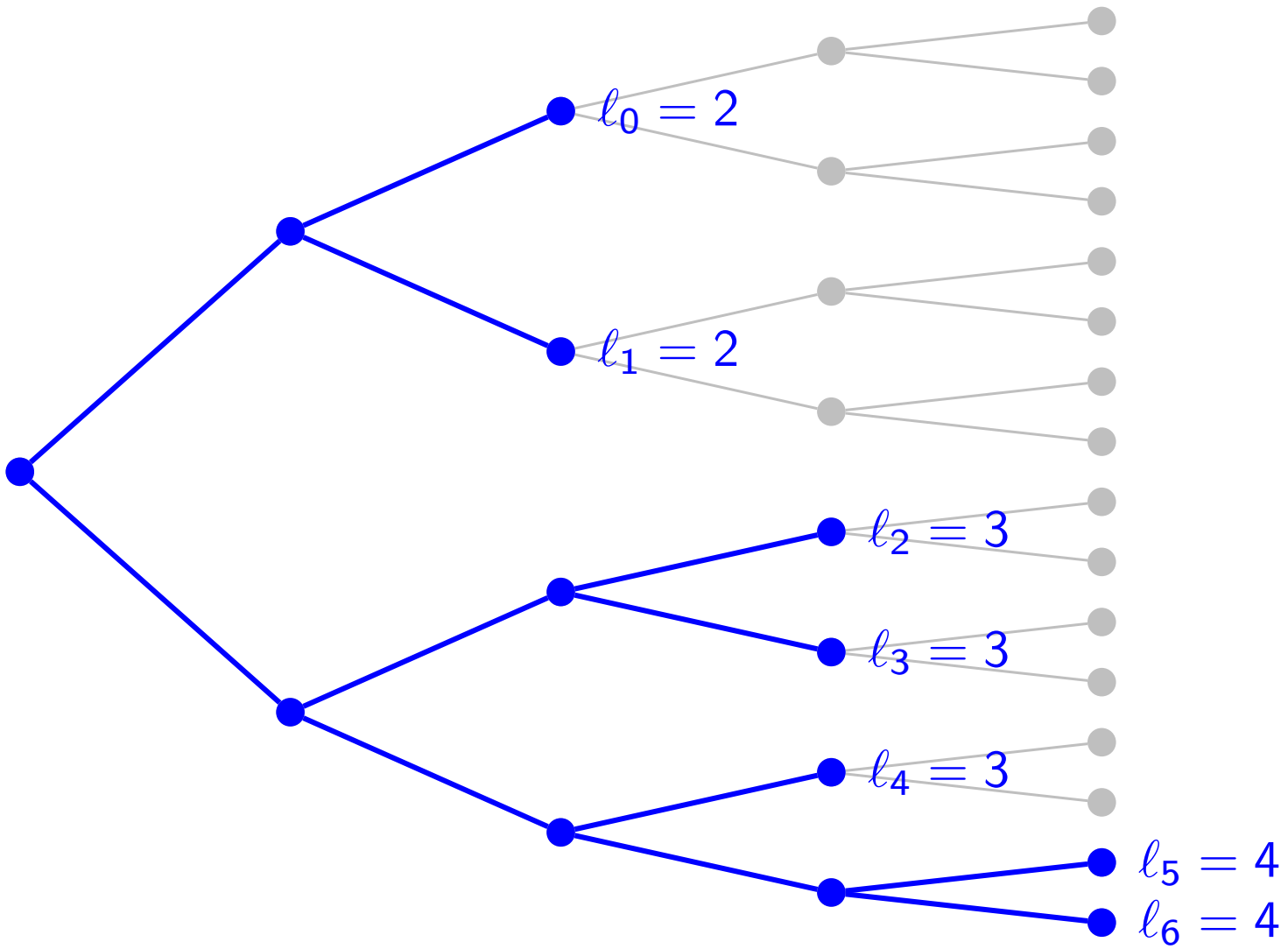
Prefix Code Construction

1. Start with balanced tree of infinite depth (or maximum depth)
2. Init codeword length index $k = 0$
3. Choose any node of depth ℓ_k and prune tree at this node
4. Increment codeword length index $k = k + 1$
5. If there more codeword lengths, proceed with 3

Prefix Code Construction Example

k	ℓ_k
0	**2**
1	**2**
2	**3**
3	**3**
4	**3**
5	**4**
6	**4**

$$\sum_{\forall k} 2^{-\ell_k} = 1$$



Is This Code Construction Always Possible ?

- Selection of a node at depth ℓ_k removes $2^{\ell_i - \ell_k}$ choices at depth $\ell_i \geq \ell_k$

➡ Remaining choices $n(\ell_i)$ at depth $\ell_i \geq \ell_k$ are given by

$$n(\ell_i) = 2^{\ell_i} - \sum_{\forall k<i} 2^{\ell_i - \ell_k} = 2^{\ell_i} \cdot 1 - \sum_{\forall k<i} 2^{\ell_i - \ell_k} \tag{12}$$

- Using the given condition $\zeta(\gamma) = \sum_{\forall k} 2^{-\ell_k} \leq 1$ yields

$$\begin{aligned} n(\ell_i) &\geq 2^{\ell_i} \left(\sum_{\forall k} 2^{-\ell_k} \right) - \sum_{\forall k<i} 2^{\ell_i - \ell_k} = \sum_{\forall k \geq i} 2^{\ell_i - \ell_k} \\ &= 2^{\ell_i - \ell_i} + \sum_{\forall k>i} 2^{\ell_i - \ell_k} = 1 + \sum_{\forall k>i} 2^{\ell_i - \ell_k} \end{aligned}$$

➡ $$n(\ell_i) \geq 1 \tag{13}$$

➡ **Can always construct prefix code for $\zeta(\gamma) \leq 1$**

Kraft Inequality

Necessary Condition for Uniquely Decodable Codes (Kraft Inequality)

- For each uniquely decodable code, the set of codeword lengths fulfills

$$\zeta(\gamma) = \sum_{\forall k} 2^{-\ell_k} \leq 1 \tag{14}$$

Proof of Kraft Inequality

- For a given symbol alphabet $\mathcal{A} = \{a_0, a_1, a_2, \cdots\}$, we want to prove

$$\sum_{\forall a \in \mathcal{A}} 2^{-\ell(a)} \leq 1 \tag{15}$$

- Trick: Consider L-th power of the left side

$$\left(\sum_{\forall a \in \mathcal{A}} 2^{-\ell(a)} \right)^L = \sum_{\forall x_0 \in \mathcal{A}} \sum_{\forall x_1 \in \mathcal{A}} \cdots \sum_{\forall x_{K-1} \in \mathcal{A}} 2^{-\ell(x_0)} \cdot 2^{-\ell(x_1)} \cdot \ldots \cdot 2^{-\ell(x_{L-1})} \tag{16}$$

Proof of Kraft Inequality (continued)

$$\left(\sum_{\forall a \in \mathcal{A}} 2^{-\ell(a)}\right)^{L} = \sum_{\forall x_0 \in \mathcal{A}} \sum_{\forall x_1 \in \mathcal{A}} \cdots \sum_{\forall x_{L-1} \in \mathcal{A}} 2^{-(\ell(x_0)+\ell(x_1)+\cdots+\ell(x_{L-1}))}$$

$$= \sum_{\forall \boldsymbol{x}^{(L)} \in \mathcal{A}^L} 2^{-\ell(\boldsymbol{x}^{(L)})} \tag{17}$$

- Interpretion:
  - Sum over all possible symbol sequences $\boldsymbol{x}^{(L)}$ with L symbols
  - $\ell(\boldsymbol{x}^{(L)})$ is the combined codeword length for the symbol sequence $\boldsymbol{x}^{(L)}$

➡ Rewrite using codeword lengths $\ell^{(L)}$ for L symbols

$$\left(\sum_{\forall a \in \mathcal{A}} 2^{-\ell(a)}\right)^{L} = \sum_{\forall \boldsymbol{x}^{(L)} \in \mathcal{A}^L} 2^{-\ell(\boldsymbol{x}^{(L)})} = \sum_{\ell^{(L)}=1}^{L \cdot \ell_{\max}} n\left(\ell^{(L)}\right) \cdot 2^{-\ell^{(L)}} \tag{18}$$

where

- $\ell_{\max}$ is the longest codeword length for a single symbol
- $n\left(\ell^{(k)}\right)$ is the number of source sequences $\boldsymbol{x}^{(L)}$ with L symbols that are represented by a combined codeword of length $\ell^{(L)}$

Proof of Kraft Inequality (continued)

- There are only $2^{\ell^{(L)}}$ distinct bit sequences of length $\ell^{(L)}$

➡ For a uniquely decodable code, we require $n\left(\ell^{(L)}\right) \leq 2^{\ell^{(L)}}$, yielding

$$\left(\sum_{\forall a \in \mathcal{A}} 2^{-\ell(a)}\right)^{L} = \sum_{\ell^{(L)}=1}^{L \cdot \ell_{\max}} n\left(\ell^{(L)}\right) \cdot 2^{-\ell^{(L)}} \leq \sum_{\ell^{(L)}=1}^{L \cdot \ell_{\max}} 2^{\ell^{(L)} - \ell^{(L)}} = \sum_{\ell^{(L)}=1}^{L \cdot \ell_{\max}} 1 = L \cdot \ell_{\max} \tag{19}$$

➡ Hence, we require

$$\sum_{\forall a \in \mathcal{A}} 2^{-\ell(a)} \leq \sqrt[L]{L \cdot \ell_{\max}} \tag{20}$$

➡ Must be true for arbitrarily long sequences, $L \to \infty$, yielding

$$\sum_{\forall a \in \mathcal{A}} 2^{-\ell(a)} \leq \lim_{L \to \infty} \sqrt[L]{L \cdot \ell_{\max}} \tag{21}$$

Proof of Kraft Inequality (continued)

- Derive limit

$$\lim_{L\to\infty} \sqrt[L]{L \cdot \ell_{\max}} = \lim_{L\to\infty} \exp\left(\ln \sqrt[L]{L \cdot \ell_{\max}}\right) = \exp\left(\lim_{L\to\infty} \frac{\ln(L \cdot \ell_{\max})}{L}\right)$$

$$= \exp\left(\lim_{L\to\infty} \frac{\frac{\mathrm{d}}{\mathrm{d}\,L} \ln(L \cdot \ell_{\max})}{\frac{\mathrm{d}}{\mathrm{d}\,L} L}\right) = \exp\left(\lim_{L\to\infty} \frac{\frac{1}{L \cdot \ell_{\max}} \cdot \ell_{\max}}{1}\right)$$

$$= \exp\left(\lim_{L\to\infty} \frac{1}{L}\right) = \exp(0) = 1 \tag{22}$$

➡ Necessary condition for unique decodability of arbitrarily long symbol sequences

$$\sum_{\forall a \in \mathcal{A}} 2^{-\ell(a)} \leq \lim_{L\to\infty} \sqrt[L]{L \cdot \ell_{\max}}$$

$$\sum_{\forall a \in \mathcal{A}} 2^{-\ell(a)} \leq 1 \tag{23}$$

Practical Importance of Prefix Codes

We have shown

1. All uniquely decodable codes fulfill the Kraft inequality

$$\zeta(\gamma) = \sum_{\forall k} 2^{-\ell_k} \leq 1 \tag{24}$$

2. For each set of codeword lengths that fulfills the Kraft inequality, we can construct a prefix code

➡ **There are no uniquely decodable codes that have a smaller average codeword length than the best prefix code**

Prefix Codes

- Not only uniquely decodable, but also **instantaneously decodable**

➡ **All variable-length codes used in practice are prefix codes**

Kullback-Leibler Divergence (Discrete Version)

Definition

- Measure for divergence from a pmf q to a pmf p

$$D(p||q) = \sum_{\forall k} p_k \log_2 \left(\frac{p_k}{q_k} \right) \tag{25}$$

- Note, in general we have

$$D(p||q) \neq D(q||p) \tag{26}$$

Divergence Inequality

- Divergence is non-negative,

$$D(p||q) = \sum_{\forall k} p_k \log_2 \left(\frac{p_k}{q_k} \right) \geq 0 \tag{27}$$

with equality if and only if $p = q \quad (\forall k, p_k = q_k)$

Proof of Divergence Inequality

- Definition of divergence

$$D(p||q) = \sum_{\forall k} p_k \log_2\left(\frac{p_k}{q_k}\right) = -\frac{1}{\ln 2}\sum_{\forall k} p_k \ln\left(\frac{q_k}{p_k}\right) \tag{28}$$

- Use inequality $\ln x \leq x - 1$ (with equality if and only if $x = 1$),

$$\begin{aligned} D(p||q) &= -\frac{1}{\ln 2}\sum_{\forall k} p_k \ln\left(\frac{q_k}{p_k}\right) && \text{(apply: } -\ln x \geq 1 - x\text{)} \\ &\geq \frac{1}{\ln 2}\sum_{\forall k} p_k \left(1 - \frac{q_k}{p_k}\right) && \text{(equality: } \forall k, p_k = q_k\text{)} \\ &= \frac{1}{\ln 2}\left(\sum_{\forall k} p_k - \sum_{\forall k} q_k\right) = \frac{1}{\ln 2}(1-1) = 0 \end{aligned}$$

$$\rightarrow \quad D(p||q) \geq 0 \qquad \text{(equality: } p = q\text{)} \tag{29}$$

Lower Bound for Average Codeword Length

- Re-write definition of average codeword length

$$\bar{\ell} = \sum_{\forall k} p_k \, \ell_k = \left(\sum_{\forall k} p_k \, \ell_k \right) + \log_2 \left(\sum_{\forall i} 2^{-\ell_i} \right) - \log_2 \left(\sum_{\forall i} 2^{-\ell_i} \right) \tag{30}$$

- Applying Kraft inequality (in logarithmic form),

$$- \log_2 \left(\sum_{\forall i} 2^{-\ell_i} \right) \geq 0,$$

yields

$$\bar{\ell} \geq \left(\sum_{\forall k} p_k \, \ell_k \right) + \log_2 \left(\sum_{\forall i} 2^{-\ell_i} \right) \tag{31}$$

Lower Bound for Average Codeword Length

- Re-write expression

$$
\begin{aligned}
\bar{\ell} \;&\geq\; \left(\sum_{\forall k} p_k \, \ell_k \right) + \log_2 \left(\sum_{\forall i} 2^{-\ell_i} \right) \\
&=\; \left(\sum_{\forall k} p_k \, \ell_k \right) + \left(\sum_{\forall k} p_k \right) \log_2 \left(\sum_{\forall i} 2^{-\ell_i} \right) \\
&=\; \sum_{\forall k} p_k \, \left(\ell_k + \log_2 \left(\sum_{\forall i} 2^{-\ell_i} \right) \right) \\
&=\; \sum_{\forall k} p_k \, \left(-\log_2 \left(2^{-\ell_k} \right) + \log_2 \left(\sum_{\forall i} 2^{-\ell_i} \right) \right) \\
&=\; -\sum_{\forall k} p_k \, \log_2 \left(\frac{2^{-\ell_k}}{\sum_{\forall i} 2^{-\ell_i}} \right)
\end{aligned}
\tag{32}
$$

Lower Bound for Average Codeword Length

- Define new pmf q, with probability masses

$$q_k = \frac{2^{-\ell_k}}{\sum_{\forall i} 2^{-\ell_i}} \qquad \left(\text{note:} \quad \sum_{\forall k} q_k = 1 \right) \tag{33}$$

- We can then write

$$\begin{aligned} \bar{\ell} \geq -\sum_{\forall k} p_k \log_2 q_k &= -\sum_{\forall k} p_k \left(\log_2 q_k + \log_2 p_k - \log_2 p_k \right) \\ &= -\sum_{\forall k} p_k \log_2 p_k + \sum_{\forall k} p_k \log_2 \left(\frac{p_k}{q_k} \right) \\ &= -\sum_{\forall k} p_k \log_2 p_k + D(p||q) \end{aligned} \tag{34}$$

- Applying the divergence inequality, $D(p||q) \geq 0$, finally yields

$$\boxed{\bar{\ell} \geq -\sum_{\forall k} p_k \log_2 p_k} \tag{35}$$

Entropy and Redundany

Entropy

- **Entropy of a random variable** S with pmf p (and alphabet $\mathcal{A}$)

$$H(p) = H(S) = \mathrm{E}\{-\log_2 p_S(S)\} = -\sum_{\forall a \in \mathcal{A}} p_S(a) \log_2 p_S(a) = -\sum_{\forall k} p_k \log_2 p_k \tag{36}$$

➡ Measure for uncertainty about a random variable S (with pmf p)

➡ **Lower bound for average codeword length** (of a uniquely decodable code)

$$\bar{\ell}(\gamma) \geq H(p) \tag{37}$$

Redundancy: Measure for Efficiency of a Code

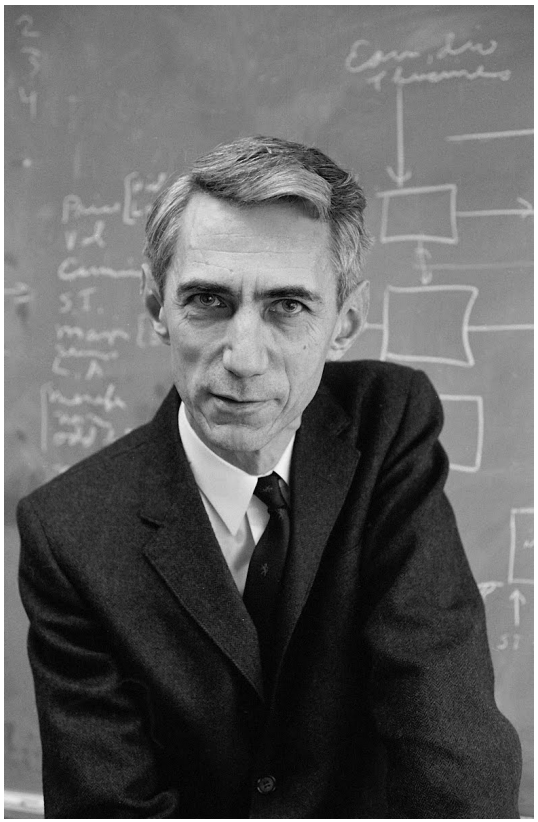
- Redundancy of a lossless code γ for pmf p

$$\varrho(\gamma) = \bar{\ell}(\gamma) - H(p) \geq 0 \tag{38}$$

- Often used as relative measure $\varrho(\gamma)/H(p)$

Historical Reference

- SHANNON introduced entropy as an uncertainty measure for random experiments and derived it based on three postulates
- Published 1 year later as "The Mathematical Theory of Communication"
- Founding work of the field of "Information Theory"



The Bell System Technical Journal

Vol. XXVII *July, 1948* *No. 3*

A Mathematical Theory of Communication

By C. E. SHANNON

INTRODUCTION

THE recent development of various methods of modulation such as PCM and PPM which exchange bandwidth for signal-to-noise ratio has intensified the interest in a general theory of communication. A basis for such a theory is contained in the important papers of Nyquist[1] and Hartley[2] on this subject. In the present paper we will extend the theory to include a number of new factors, in particular the effect of noise in the channel, and the savings possible due to the statistical structure of the original message and due to the nature of the final destination of the information.

Prefix Codes with Zero Redundancy

Used two inequalities in derivation

1 Kraft inequality

$$\sum_{\forall k} 2^{-\ell_k} \leq 1$$

➡ Equality if prefix code represents a full binary tree (always possible),

$$\bar{\ell} = H(p) + D(p||q) \qquad \text{with} \qquad q_k = 2^{-\ell_k} \tag{39}$$

2 Divergence inequality

$$D(p||q) \geq 0 \qquad (\text{equality for } p_k = q_k, \forall k)$$

➡ Equality if additionally all codeword lengths are given by

$$\ell_k = -\log_2 p_k$$

➡ **Zero redundancy codes are only possible if all probability masses for the source represent negative integer powers of two**

Upper Bound for Achievable Average Codeword Length

Shannon Coding

- Introduced to prove Shannon's lossless coding theorem
- For a given pmf p, all codeword lengths ℓ_k are chosen according to

$$\ell_k = \lceil -\log_2 p_k \rceil \tag{40}$$

Uniquely decodable?

- Codeword lengths satisfy Kraft inequality (use $\lceil x \rceil \geq x$)

$$\sum_{\forall k} 2^{-\ell_k} = \sum_{\forall k} 2^{-\lceil -\log_2 p_k \rceil} \leq \sum_{\forall k} 2^{\log_2 p_k} = \sum_{\forall k} p_k = 1 \tag{41}$$

Obtained Average Codeword Length

- Average codeword length $\bar{\ell}$ satisfies (use $\lceil x \rceil < x + 1$)

$$\bar{\ell} = \sum_{\forall k} p_k \, \ell_k = \sum_{\forall k} p_k \lceil -\log_2 p_k \rceil < \sum_{\forall k} p_k \left(1 - \log_2 p_k\right) = H(p) + 1 \tag{42}$$

Bounds on Average Codeword Length

Uniquely decodable codes

- Lower bound on average codeword length of all uniquely decodable codes

$$\bar{\ell} \geq H(S)$$

with $H(S)$ being the entropy of the random variables S

$$H(S) = H(p) = -\sum_{\forall k} p_k \log_2 p_k$$

- Equality can only be achieved if all probability masses represent negative integer powers of two

Minimum average codeword length

- Minimum achievable average codeword length for a source is bounded by

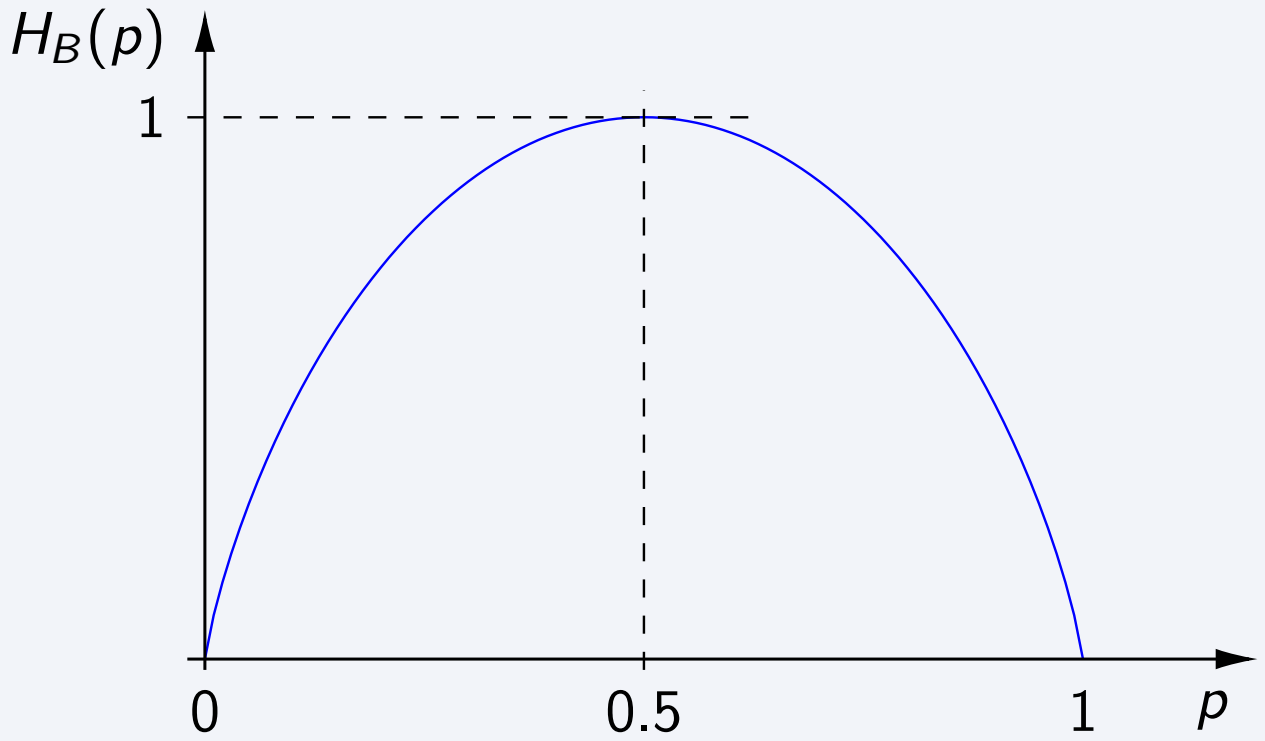
$$H(S) \leq \bar{\ell}_{\text{min}} < H(S) + 1$$

Binary Entropy Function

Binary Source

- Probability mass function: $\{p, 1-p\}$
- Entropy

$$H(S) = H_B(p) = -p \log_2 p - (1-p) \log_2 (1-p) \tag{43}$$



Optimal Lossless Codes

Optimal Prefix Codes

- **Optimal prefix code**: Any prefix code that achieves the minimum possible average codeword length for the given pmf

$$\bar{\ell} = \sum_{\forall a \in \mathcal{A}} p(a) \cdot \ell(a) = \sum_{\forall k} p_k \cdot \ell_k$$

➡ Note: Each optimal prefix code is also an optimal uniquely decodable code

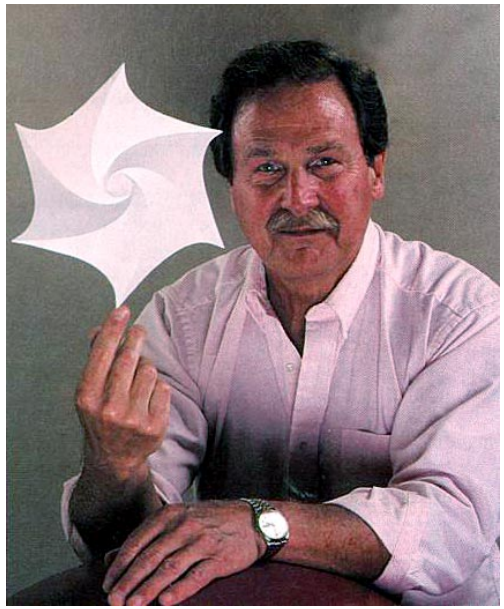
How can we construct an optimal prefix code?

- Answer was given in 1952:

  David A. Huffman: "A Method for the Construction of Minimum-Redundancy Codes"

- Finite symbols alphabets:

  Huffman algorithm always finds a prefix code with minimum redundancy



Exchanging Codewords

Impact on Average Codeword Length

Consider a prefix code with an average codeword length $\bar{\ell}$

- Two alphabet letters a and b, with probabilities p_a and p_b, have codewords of lengths ℓ_a and ℓ_b
- If we exchange the codewords for the alphabet letters a and b, we obtain another prefix code with an averge codeword length $\bar{\ell}'$ given by

$$\begin{aligned} \bar{\ell}' &= \bar{\ell} - (p_a \ell_a + p_b \ell_b) + (p_a \ell_b + p_b \ell_a) \\ &= \bar{\ell} + p_a(\ell_b - \ell_a) - p_b(\ell_b - \ell_a) \\ &= \bar{\ell} + (p_a - p_b)(\ell_b - \ell_a) \end{aligned} \tag{44}$$

Following cases:

- ➡ $p_a > p_b$ and $\ell_a > \ell_b$: Decrease of average codeword length
- ➡ $p_a > p_b$ and $\ell_a < \ell_b$: Increase of average codeword length
- ➡ $p_a = p_b$ or $\ell_a = \ell_b$: No modification of average codeword length

Subset of Optimal Prefix Codes

Lemma

For any finite symbol alphabet $\mathcal{A}$, there exists an optimal prefix code $\mathcal{C}$ such that the two longest codewords have the same length, differ only in the final bit, and correspond to the two least likely alphabet letters.

Proof

1. *For each codeword of maximum length, the code includes a codeword of the same length that differs only in the final bit*:

   In the binary tree representation, this means that the corresponding terminal node has a sibling.

   If this were not the case, the code $\mathcal{C}$ cannot be optimal, since a removal of the last bit of the considered codeword of maximum length would reduce the average codeword length without violating the prefix property.

(*continued on next slide*)

Subset of Optimal Prefix Codes

Proof (continued)

2 *Two of the codewords of maximum length correspond to the two least likely alphabet letters*:

For any two alphabet letters with $p_a > p_b$, the codeword length must satisfy $\ell_a \leq \ell_b$. Otherwise, an exchange of the codewords would reduce the average codeword length without violating the prefix property (as shown above).

Note: For two alphabet letters with $p_a = p_b$, an exchange of the codewords does not modify the average codeword length (as shown above).

3 *Two codewords with maximum length that differ only in the last bit correspond to the two least likely alphabet letters*:

This condition is not necessary for optimality, but we can exchange any two codewords of the same length (maximum length) without impacting the average codeword length (see above) or the prefix property.

Hence, there exist optimal prefix codes with this property.

The Huffman Algorithm

General Idea of Huffman Algorithm

- Consider subset of optimal prefix codes for which the two least likely symbols correspond to codewords of maximum length that differ only in the final bit

➡ Choose the two least likely symbols and create a parent node in code tree

➡ Repeat the procedure with a reduced alphabet

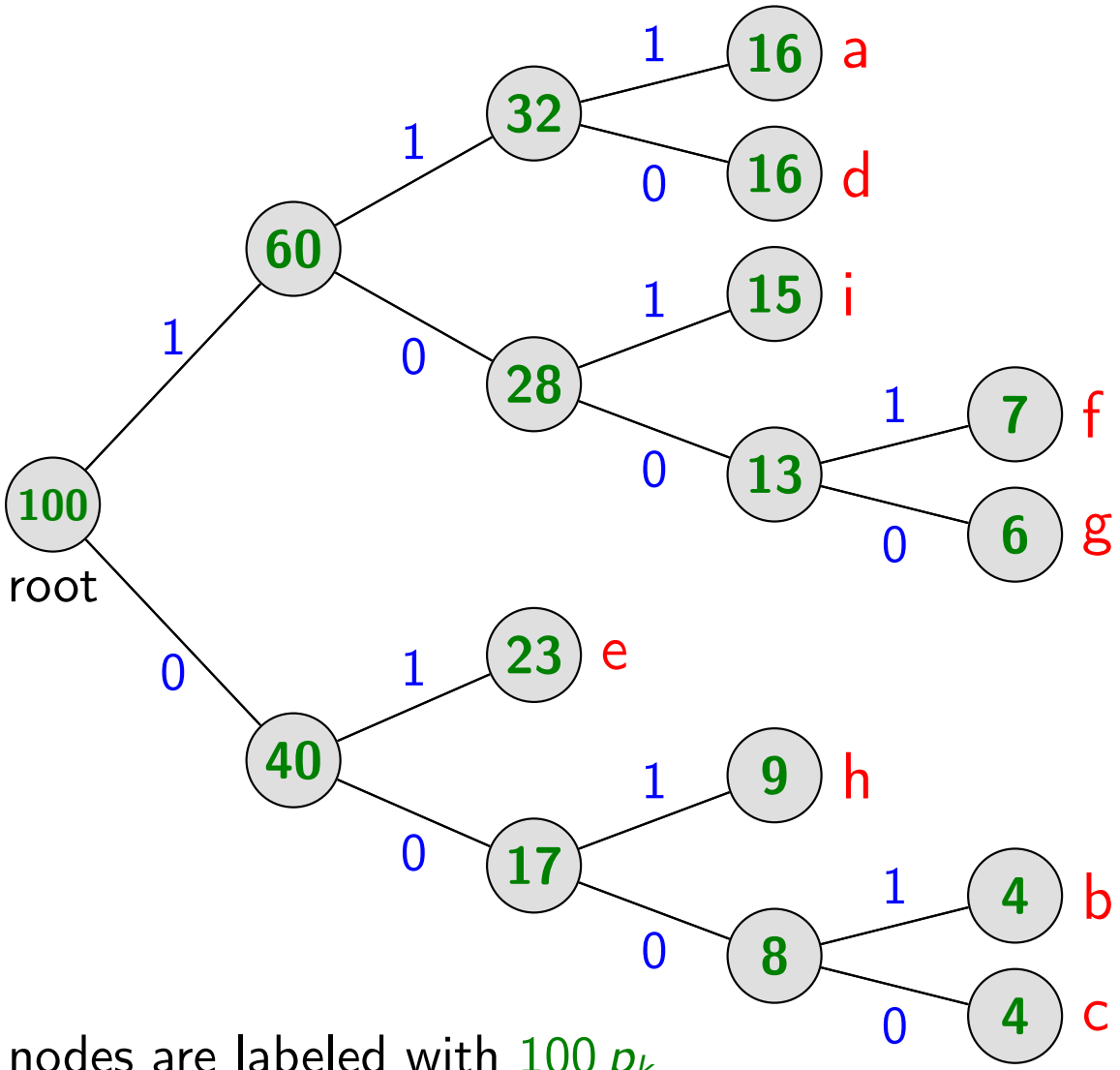
Huffman Algorithm (via construction of binary code tree)

1. Select the two letters a and b with the smallest probabilities p_a and p_b
2. Create a parent node for the two letters a and b in the binary code tree
3. Replace the letters a and b with a new letter with a probability $p_a + p_b$
4. If the resulting new alphabet contains more than a single letter, repeat all previous steps with this alphabet
5. Convert the obtained binary code tree into a prefix code

Example: Construction of a Huffman Code

$$\begin{aligned} \bar{\ell} &= 2.98 \\ H(p) &\approx 2.9405 \\ \varrho &\approx 0.0395 \ (1.34\%) \end{aligned}$$

a_k	p_k	$\boldsymbol{b}_k$
a	0.16	111
b	0.04	0001
c	0.04	0000
d	0.16	110
e	0.23	01
f	0.07	1001
g	0.06	1000
h	0.09	001
i	0.15	101



note: nodes are labeled with 100 p_k

Remarks on Huffman Codes

Optimality of Huffman codes

- All codes obtained by Huffman algorithm are optimal prefix codes
- Rigorous proof (by induction) can be found in
  [T. M. Cover, J. A. Thomas, "Elements of Information Theory"]
- There are multiple Huffman codes
  - Large degree of freedom for labelling tree branches with "0" and "1"
  - There may exist different possibilites for creating the next parent node (if multiple symbols have the probability)

Other optimal prefix codes

- Huffman algorithm creates prefix codes of a special class
- There might be optimal codes that cannot be obtained by the Huffman algorithm (see exercises)

Important: Huffman algorithm always yields one optimal prefix code

Examples for Huffman Codes: MPEG-2 Video

Coding of luma DC transform coefficient

Table B.12 – Variable length codes for dct_dc_size_luminance

Variable length code	**dct_dc_size_luminance**
100	0
00	1
01	2
101	3
110	4
1110	5
1111 0	6
1111 10	7
1111 110	8
1111 1110	9
1111 1111 0	10
1111 1111 1	11

Summary

Prefix Codes

- Binary code trees / simple decoding process
- Instantaneous codes (suitable for complicated syntax)

Unique Decodability

- Necessary condition: Kraft inequality for codeword lengths: $\sum_k 2^{-\ell_k} \leq 1$
- Can always construct prefix code for codeword length $\{\ell_k\}$ with $\sum_k 2^{-\ell_k} \leq 1$

➡ **All uniquely decodable codes used in practice are prefix codes**

Entropy

- Lower bound $H(p)$ for average codeword length (given by pmf p)
- Only achievable if all probability masses are negative integer powers of two
- Can always construct code with $H(p) \leq \bar{\ell} < H(p) + 1$

Huffman Algorithm / Huffman Codes

- Algorithm for construction of optimal prefix codes (Huffman codes)
- Not all optimal codes can be constructed by Huffman algorithm