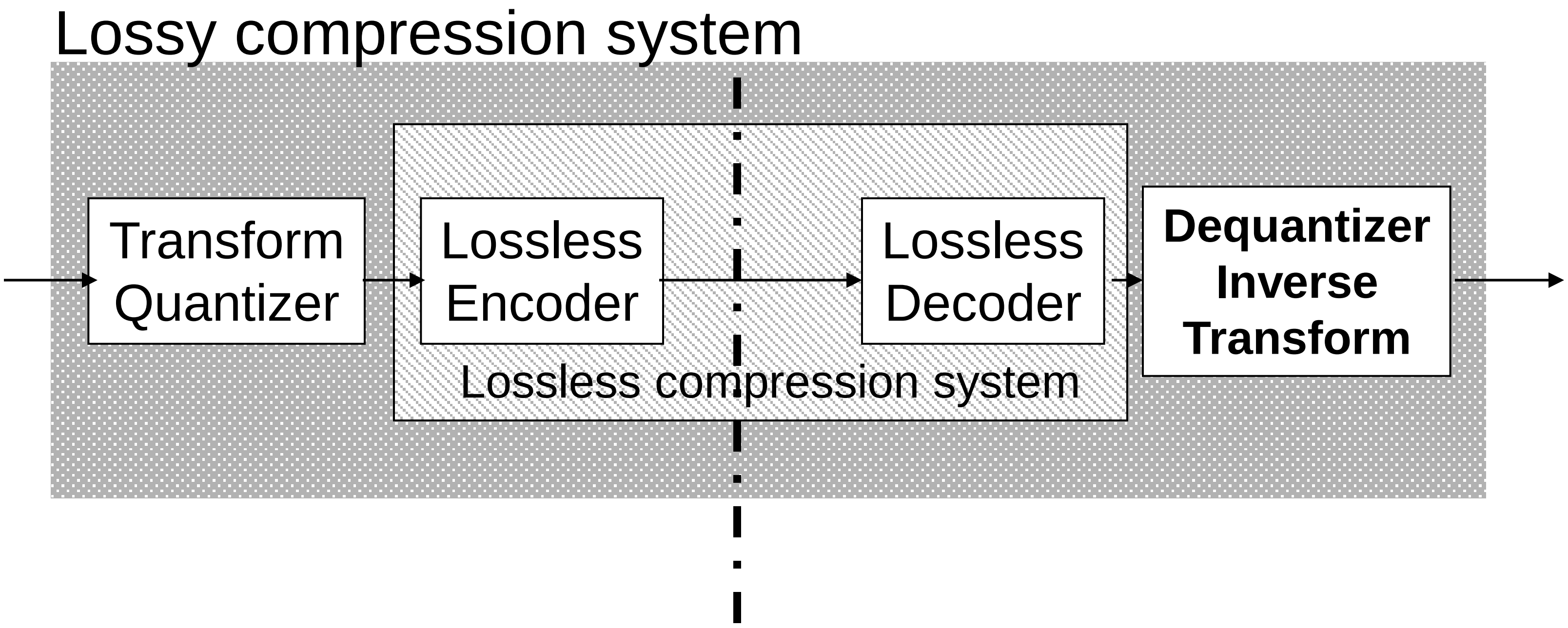
Lossless compression in lossy compression systems

- Almost every lossy compression system contains a lossless compression system



- We discuss the basics of lossless compression first, then move on to lossy compression

Topics in lossless compression

- Binary decision trees and variable length coding
- Entropy and bit-rate
- Prefix codes, Huffman codes, Golomb codes

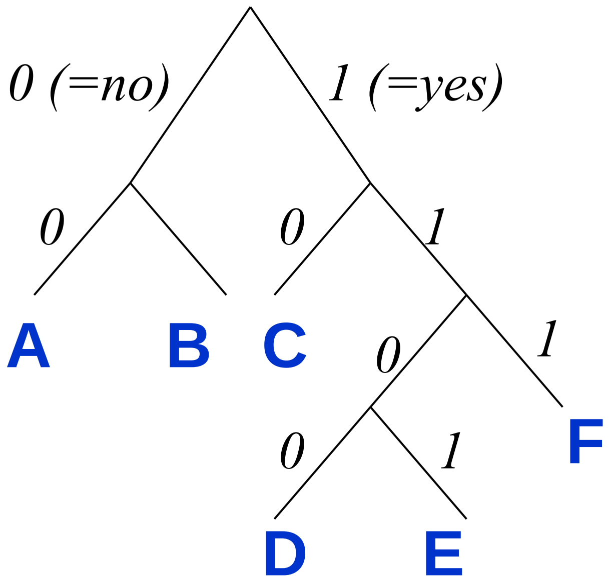
- Joint entropy, conditional entropy, sources with memory
- Fax compression standards
- Arithmetic coding

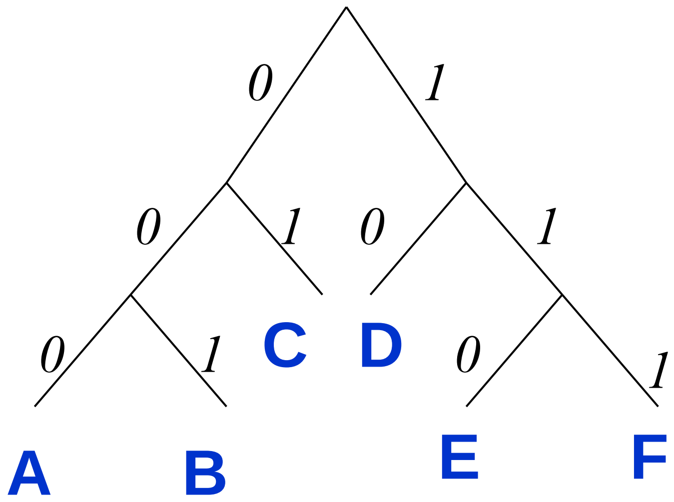
Example: 20 Questions

- *Alice* thinks of an outcome (from a finite set), but does not disclose her selection.
- *Bob* asks a series of yes/no questions to uniquely determine the outcome chosen. The goal of the game is to ask as few questions as possible on average.
- Our goal: Design the best strategy for *Bob.*

Example: 20 Questions (cont.)

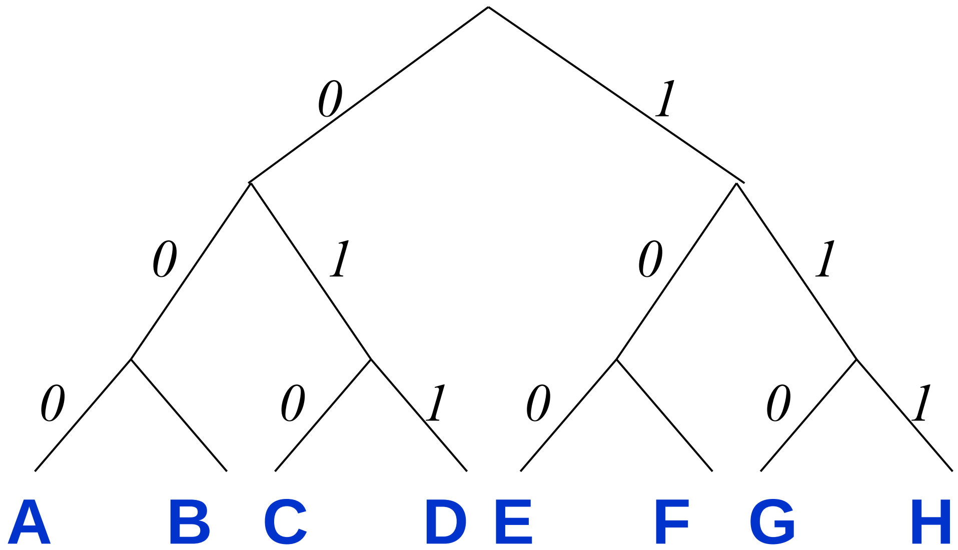
- Which strategy is better?





- Observation: The collection of questions and answers yield a binary code for each outcome.

Fixed length codes



- Average description length for K outcomes $l_{av} = \log_2 K$
- Optimum for equally likely outcomes
- Verify by modifying tree

Variable length codes

- If outcomes are NOT equally probable:
  - Use shorter descriptions for likely outcomes
  - Use longer descriptions for less likely outcomes
- Intuition:
  - Optimum balanced code trees, i.e., with equally likely outcomes, can be pruned to yield unbalanced trees with unequal probabilities.
  - The unbalanced code trees such obtained are also optimum.
  - Hence, an outcome of probability p should require about

$$\log_2\left(\frac{1}{p}\right) \text{ bits}$$

Entropy of a random variable

- Consider a discrete, finite-alphabet random variable X

$$\text{Alphabet } \mathcal{A}_X = \{\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_{K-1}\}$$
$$\text{PMF } \; f_X(x) = P(X = x) \quad \text{for each } x \in \mathcal{A}_X$$

- **Information** associated with the event $X=x$

$$h_X(x) = -\log_2 f_X(x)$$

- **Entropy of *X*** is the expected value of that information

$$H(X) = E\left[h_X(X)\right] = -\sum_{x \in \mathcal{A}_X} f_X(x) \log_2 f_X(x)$$

- Unit: bits

Information and entropy: properties

- Information $h_X(x) \geq 0$
- Information $h_X(x)$ strictly increases with decreasing probability $f_X(x)$
- Boundedness of entropy

$$0 \leq H(X) \leq \log_2\left(\left\|\mathcal{A}_X\right\|\right)$$

Equality if only one outcome can occur

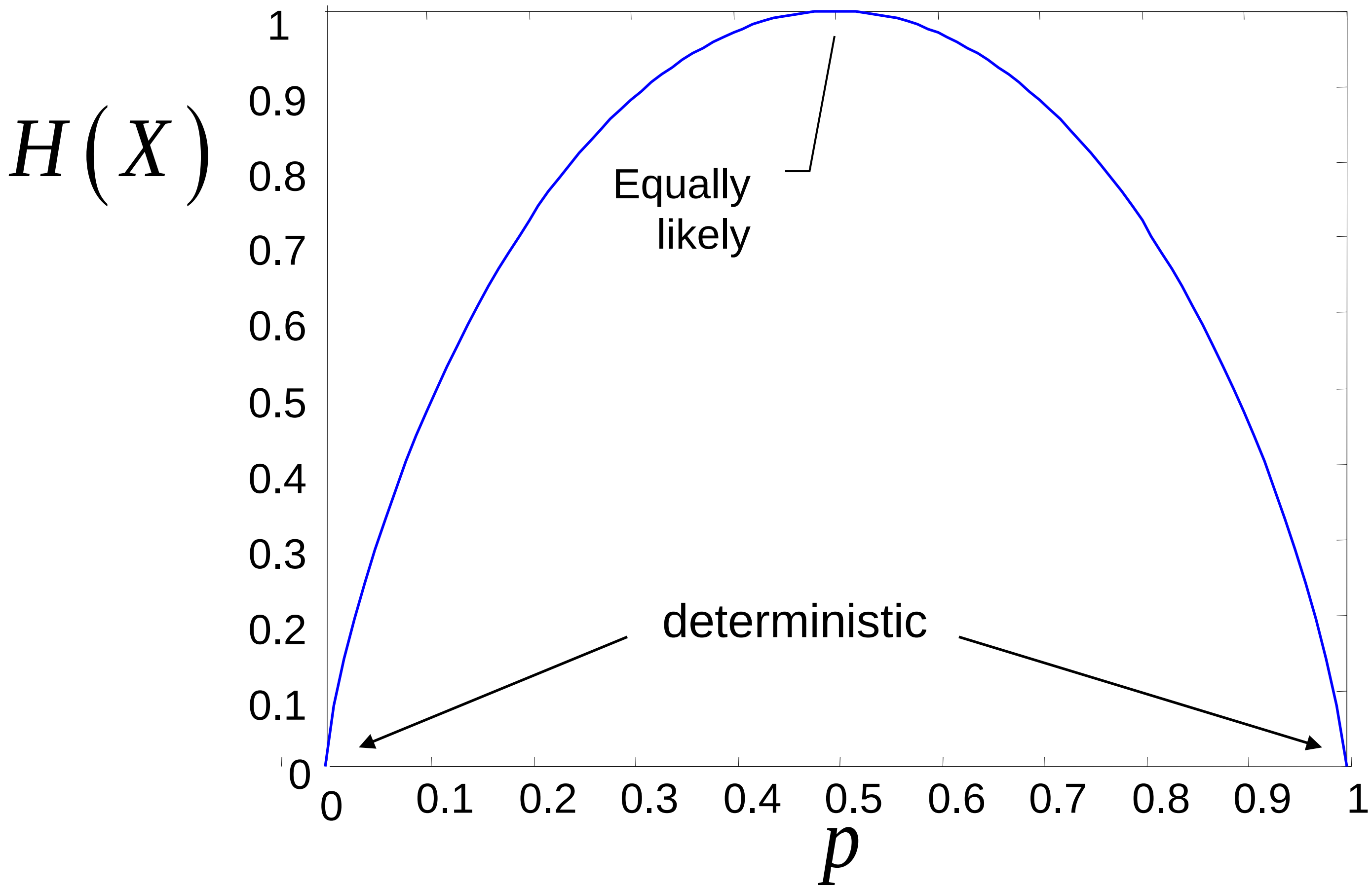
Equality if all outcomes are equally likely

- Very likely and very unlikely events do not substantially change entropy

$$-p\log_2 p \to 0 \quad \text{for } p \to 0 \text{ or } p \to 1$$

Example: Binary random variable

$$H(X) = -p\log_2 p - (1-p)\log_2(1-p)$$



Entropy and bit-rate

- Consider IID random process $\{X_n\}$ (or "source") where each sample X_n (or "symbol") possesses identical entropy $H(X)$
- $H(X)$ is called "entropy rate" of the random process.
- Noiseless Source Coding Theorem *[Shannon, 1948]*
  - The entropy $H(X)$ is a lower bound for the average word length R of a decodable variable-length code for the symbols.
  - Conversely, the average word length R can approach $H(X)$, if sufficiently large blocks of symbols are encoded jointly.
- Redundancy of a code:

$$\rho = R - H(X) \geq 0$$

Variable length codes

- Given IID random process $\{X_n\}$ with alphabet $\mathcal{A}_X$ and PMF $f_X(x)$
- Task: assign a distinct code word, c_x, to each element, $x \in \mathcal{A}_X$, where c_x is a string of $\|c_x\|$ bits, such that each symbol x_n can be determined, even if the codewords c_{x_n} are directly concatenated in a bitstream
- Codes with the above property are said to be "uniquely decodable."
- Prefix codes
  - No code word is a prefix of any other codeword
  - Uniquely decodable, symbol by symbol, in natural order $0, 1, 2, \ldots, n, \ldots$

Example of non-decodable code

α_i	code words
α_0	0
α_1	01
α_2	10
α_3	11

Encode sequence of source symbols $\alpha_0, \alpha_2, \alpha_3, \alpha_0, \alpha_1$

Resulting bit-stream 0 10 11 0 01

Encode sequence of source symbols $\alpha_1, \alpha_0, \alpha_3, \alpha_0, \alpha_1$

Resulting bit-stream 01 0 11 0 01

- Same bit-stream for different sequences of source symbols: ambiguous, not uniquely decodable
- BTW: Not a prefix code.

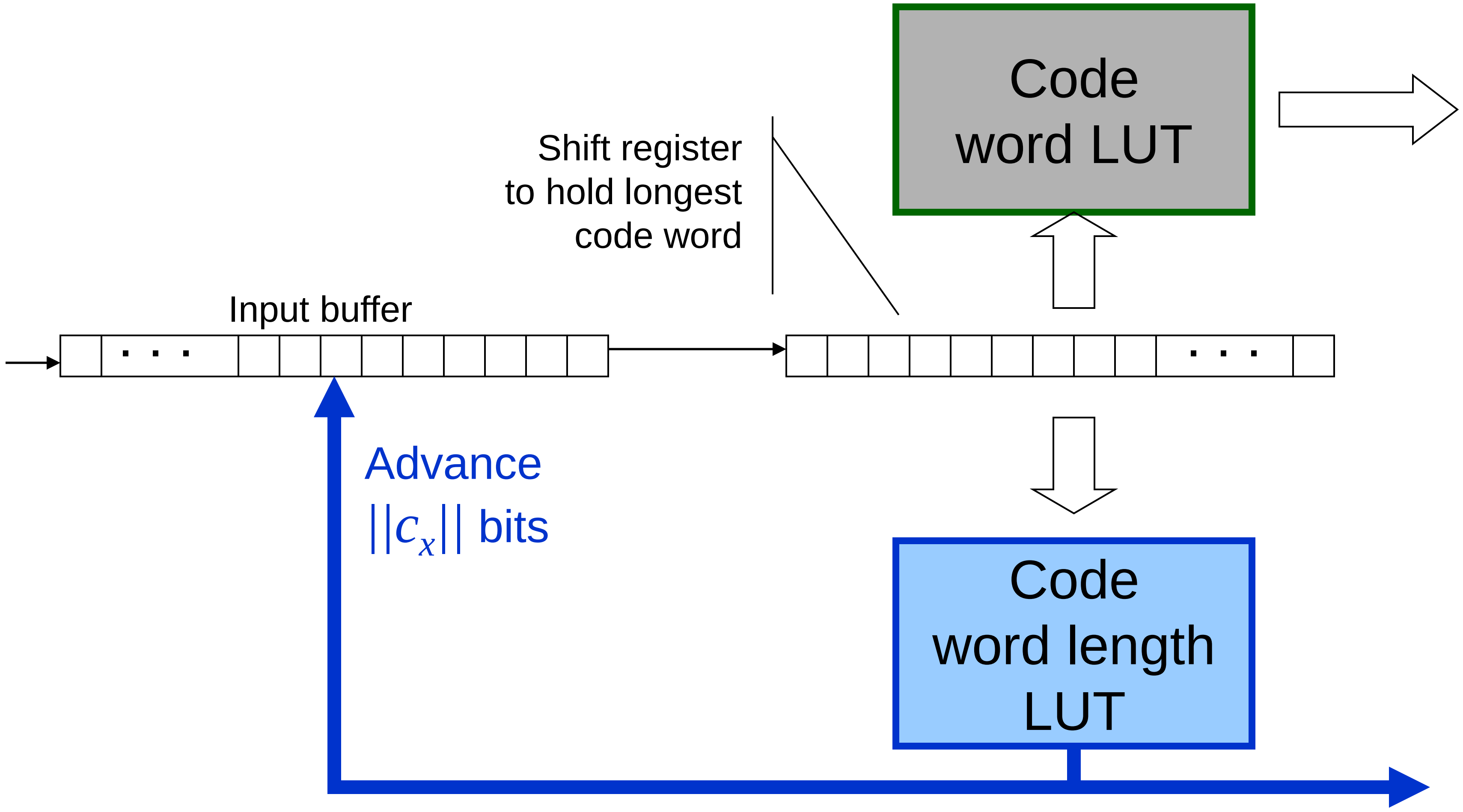
Unique decodability: McMillan and Kraft conditions

- Necessary condition for unique decodability *[McMillan]*

$$\sum_{x \in \mathcal{A}_X} 2^{-\|c_x\|} \leq 1$$

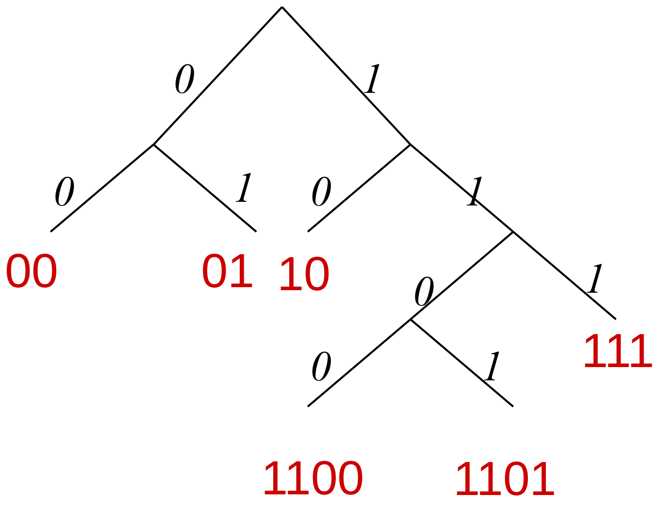
- Given a set of code word lengths $\|c_x\|$ satisfying McMillan condition, a corresponding prefix code always exists *[Kraft]*
  - Hence, McMillan inequality is both necessary and sufficient.
  - Also known as Kraft inequality or Kraft-McMillan inequality.
  - No loss by only considering prefix codes.
  - Prefix code is not unique.

Prefix Decoder



Binary trees and prefix codes

- Any binary tree can be converted into a prefix code by traversing the tree from root to leaves.



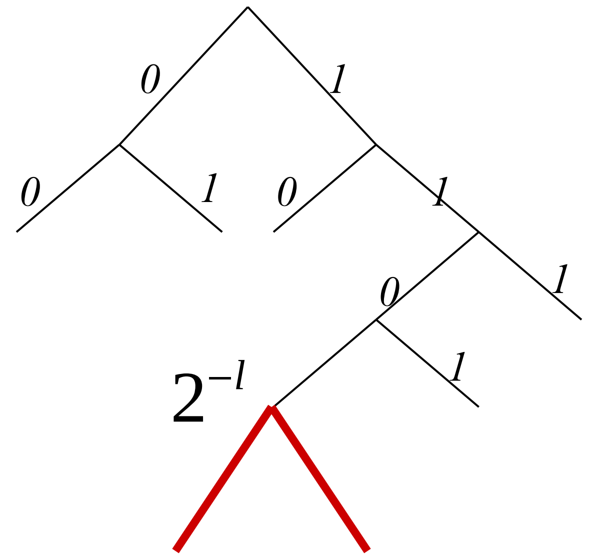
- Any prefix code corresponding to a binary tree meets McMillan condition with equality

$$3 \cdot 2^{-2} + 2 \cdot 2^{-4} + 2^{-3} = 1$$

$$\boxed{\sum_{x \in \mathcal{A}_X} 2^{-\|c_x\|} = 1}$$

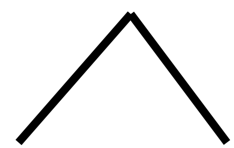
Binary trees and prefix codes (cont.)

- Augmenting binary tree by two new nodes does not change McMillan sum.
- Pruning binary tree does not change McMillan sum.



$$2^{-(l+1)} + 2^{-(l+1)} = 2^{-l}$$

- McMillan sum for simplest binary tree



$$2^{-1} + 2^{-1} = 1$$

Instantaneous variable length encoding without redundancy

- A code without redundancy, i.e.

$$R = H(X)$$

requires all individual code word lengths

$$l_{\alpha_k} = -\log_2 f_X(\alpha_k)$$

- All probabilities would have to be binary fractions:

$$f_X(\alpha_k) = 2^{-l_{\alpha_k}}$$

Example

α_i	$P(\alpha_i)$	redundant code	optimum code
α_0	0.500	00	0
α_1	0.250	01	10
α_2	0.125	10	110
α_3	0.125	11	111

$$H(X) = 1.75 \text{ bits}$$

$$R = 1.75 \text{ bits}$$

$$\rho = 0$$

Huffman Code

- Design algorithm for variable length codes proposed by Huffman (1952) always finds a code with minimum redundancy.
- Obtain code tree as follows:

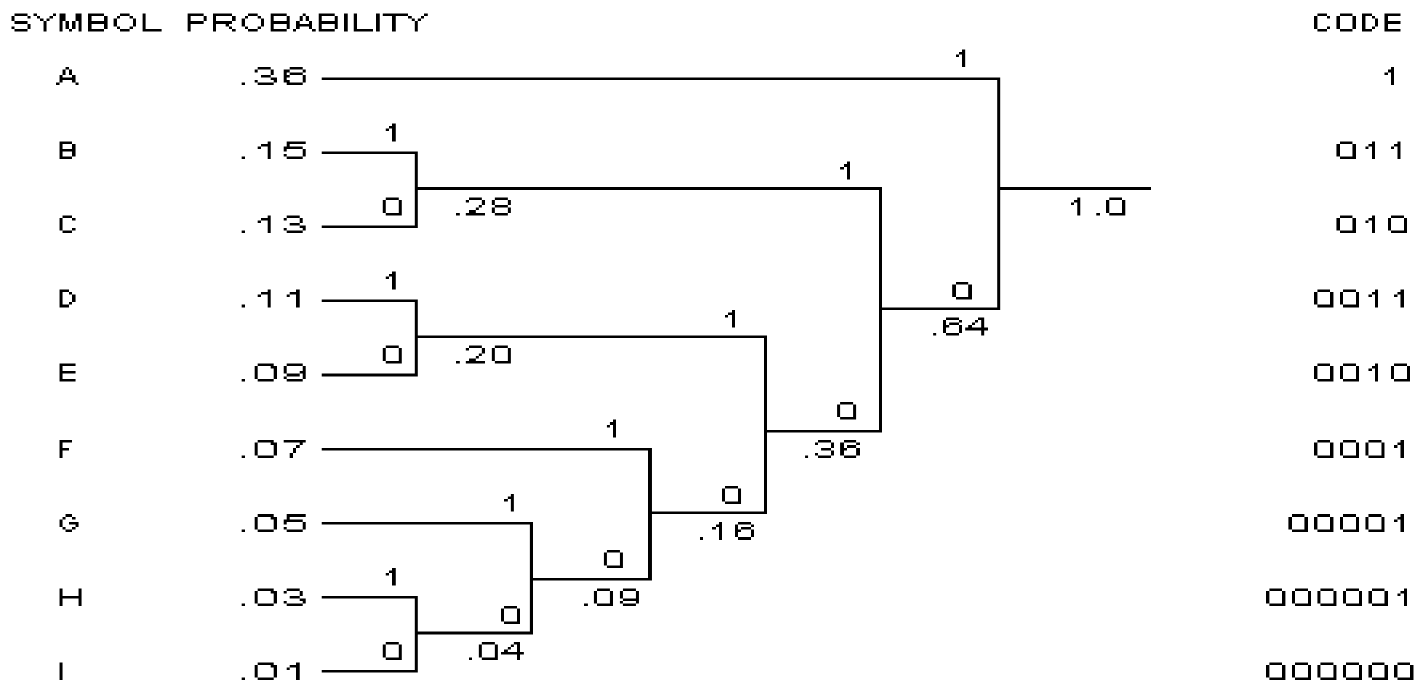
1 Pick the two symbols with lowest probabilities and merge them into a new auxiliary symbol.

2 Calculate the probability of the auxiliary symbol.

3 If more than one symbol remains, repeat steps **1** and **2** for the new auxiliary alphabet.

4 Convert the code tree into a prefix code.

Huffman Code - Example



Fixed length coding:	$R_{fixed} = 4$ bits/symbol
Huffman code:	$R_{Huffman} = 2.77$ bits/symbol
Entropy	$H(X) = 2.69$ bits/symbol
Redundancy of the Huffman code:	$\rho = 0.08$ bits/symbol

Redundancy of prefix code for general distribution

- Huffman code redundancy $0 \leq \rho < 1$ bit/symbol
- <u>Theorem:</u> For any distribution f_X, a prefix code can be found, whose rate R satisfies

$$\boxed{H(X) \leq R < H(X) + 1}$$

- Proof
  - Left hand inequality: Shannon's noiseless coding theorem
  - Right hand inequality:

Choose code word lengths $\|c_x\| = \lceil -\log_2 f_X(x) \rceil$

Resulting rate
$$R = \sum_{x \in \mathcal{A}_X} f_X(x) \lceil -\log_2 f_X(x) \rceil$$
$$< \sum_{x \in \mathcal{A}_X} f_X(x) \left(1 - \log_2 f_X(x)\right)$$
$$= H(X) + 1$$

Vector Huffman coding

- Huffman coding very inefficient for $H(X) << 1$ bit/symbol
- Remedy:
  - Combine m successive symbols to a new "block-symbol"
  - Huffman code for block-symbols
  - Redundancy

$$H(X) \leq R < H(X) + \frac{1}{m}$$

  - Can also be used to exploit statistical dependencies between successive symbols
  - Disadvantage: exponentially growing alphabet size $\|\mathcal{A}_X\|^m$

Truncated Huffman Coding

- Idea: reduce size of Huffman code table and maximum Huffman code word length by Huffman-coding only the most probable symbols.
  - Combine J least probable symbols of an alphabet of size K into an auxillary symbol ESC
  - Use Huffman code for alphabet consisting of remaining K-J most probable symbols and the symbol ESC
  - If ESC symbol is encoded, append $\lceil \log_2(J) \rceil$ bits to specify exact symbol from the full alphabet
- Results in increased average code word length – trade off complexity and efficiency by choosing J

Adaptive Huffman Coding

- Use, if source statistics are not known ahead of time
- Forward adaptation
  - Measure source statistics at encoder by analyzing entire data
  - Transmit Huffman code table ahead of compressed bit-stream
  - JPEG uses this concept (even though often default tables are transmitted)
- Backward adaptation
  - Measure source statistics both at encoder and decoder, using the same previously decoded data
  - Regularly generate identical Huffman code tables at transmitter and receiver
  - Saves overhead of forward adaptation, but usually poorer code tables, since based on past observations
  - Generally avoided due to computational burden at decoder

Unary coding

- "Geometric" source

$$\text{Alphabet } \mathcal{A}_X = \{0,1,...\} = \mathbb{Z}_+ \qquad \text{PMF} \quad f_X(x) = 2^{-(x+1)}, \quad x \geq 0$$

- Optimal prefix code with redundancy $\rho=0$ is "unary" code ("comma code")

$$c_0 = "1" \quad c_1 = "01" \quad c_2 = "001" \quad c_3 = "0001" \quad \ldots$$

- Consider geometric source with faster decay

$$\text{PMF} \quad f_X(x) = (1-\beta)\beta^x, \text{ with } 0 \leq \beta < \frac{1}{2}; \quad x \geq 0$$

- Unary code is still optimum prefix code (i.e., Huffman code), but not redundancy-free

Golomb coding

- For geometric source with slower decay

$$\text{PMF} \quad f_X(x) = (1-\beta)\beta^x, \text{ with } \frac{1}{2} < \beta < 1; \quad x \geq 0$$

- Idea: Express each *x* as

$$x = mx_q + x_r \quad \text{with} \quad x_q = \left\lfloor \frac{x}{m} \right\rfloor \quad \text{and} \quad x_r = x \bmod m$$

- Distribution of new random variables

$$f_{X_q}(x_q) = \sum_{i=0}^{m-1} f_X(mx_q + i) = \beta^{mx_q} \sum_{i=0}^{m-1} f_X(i)$$

$$f_{X_r}(x_r) = \frac{1-\beta}{1-\beta^m}\beta^{x_r} \quad \text{for} \quad 0 \leq x_r < m$$

X_q and X_r statistically independent.

Golomb coding (cont.)

- Golomb coding
  - Choose integer divisor $\beta^m \approx \frac{1}{2}$
  - Encode x_q optimally by unary code
  - Encode x_r by a modified binary code, using code word lengths

$$k_a = \lceil \log_2 m \rceil \qquad\qquad k_b = \lfloor \log_2 m \rfloor$$

  - Concatenate bits for x_q and x_r

- In practice, $m=2^k$ is often used, so x_r can be encoded by constant code word length $\log_2 m$

Golomb code examples

Unary Code

1
01
001
0001
00001
000001
0000001
00000001
000000001
.
.
.

Unary Code

10
11
010
011
0010
0011
00010
00011
000010
000011
0000010
0000011
00000010
00000011
.
.
.

Constant length code

$m=2$

100
101
110
111
0100
0101
0110
0111
00100
00101
00110
00111
000100
000101
000110
000111
.
.
.

Unary Code

Constant length code

$m=4$

Golomb parameter estimation

- Expected value for geometric distribution

$$E[X] = \sum_{x=0}^{\infty} (1-\beta)\, x\beta^x = \frac{\beta}{1-\beta} \qquad \rightarrow \qquad \beta = \frac{E[X]}{1+E[X]}$$

- Approximation for $E[X] >> 1$

$$\beta^m = \frac{\left(E[X]\right)^m}{\left(1+E[X]\right)^m} \approx 1 - \frac{m}{E[X]} \approx \frac{1}{2}$$

Rule for optimum performance of Golomb code

$$m = 2^k \approx \frac{1}{2} E[X]$$

$$k = \max\left\{0, \left\lceil \log_2\left(\frac{1}{2} E[X]\right)\right\rceil\right\}$$

Reasonable setting, even if $E[X] >> 1$ does not hold

Adaptive Golomb coder (JPEG-LS)

Initialize $A = \hat{\mu}_x$ and $N = 1$

For each $n = 0, 1, 2, \ldots$

$$\text{Set } k = \max\left\{0, \left\lceil \log_2\left(\frac{A}{2N}\right)\right\rceil\right\}$$

Code symbol x_n using Golomb code with parameter k

If $N = N_{max}$

Set $A \longleftarrow \lfloor A/2 \rfloor$ and $\mathrm{N} \longleftarrow \lfloor N/2 \rfloor$

Update $A \longleftarrow A + x_n$ and $\mathrm{N} \longleftarrow N + 1$

Initial estimate of mean

Pick the best Golomb code

Avoid overflow and slowly forget the past