ELEMENTS FROM INFORMATION THEORY

- **Any information generating process can be viewed as a source that emits a sequence of symbols chosen from a finite alphabet (for example, text: ASCII symbols; n-bit images: 2^n symbols).**

- **Simplest form of an information source:** *discrete memoryless source* **(DMS). Successive symbols produced by such a source are statistically independent.**

- **A DMS is completely specified by the source alphabet $S = \{s_1, s_2, \cdots, s_n\}$ and the associated probabilities $\{p_1, p_2, \cdots, p_n\}$.**

- *Self Information:*

$$I(s_i) = \log \frac{1}{p_i} = -\log p_i$$

  - **the occurence of a less probable event provides more information**
  - **the information of independent events taken as a single event equals the sum of the information**

- *Average Information* **per Symbol or** *Entropy* **of a DMS:**

$$H(S) = \sum_{i=1}^{n} p_i \, I(s_i) = -\sum_{i=1}^{n} p_i \, \log_2 p_i \quad \textbf{bits/symbol}$$

- **Interpretation of Entropy:**
  - **Average amount of information per symbol provided by the source (definition)**
  - **Average amount of information per symbol an observer needs to spend to remove the uncertainty in the source**

- **N-th extention of the DMS: Given a DMS of size n, group the source into blocks of N symbols. Each block can now be considered as a single source symbol generated by a source S^N with alphabet size n^N. In this case**

$$H(S^N) = N \times H(s)$$

Noiseless Source Coding Theorem

Let S be a source with alphabet size n and entropy $H(S)$. Consider coding blocks of N source symbols into binary codewords. For any $\delta > 0$, it is possible by choosing N large enough to construct a code in such a way that the average number of bit per original source symbol l_{avg} satisfies

$$H(S) \leq l_{avg} < H(S) + \delta$$

<u>"GOOD" CODES</u>

Fixed-length codes:
example: ASCII code: a− >1000011, A− >1000001

Variable-length codes:
example: Morse code (mid-19th century): e(.), a(.-), q(–.-)

Example

Symbols	Probability	Code 1	Code 2	Code 3	Code 4
s_1	1/2	0	0	0	0
s_2	1/4	0	1	10	01
s_3	1/8	1	00	110	011
s_4	1/8	10	11	111	0111
Average length		1.125	1.25	1.75	1.875

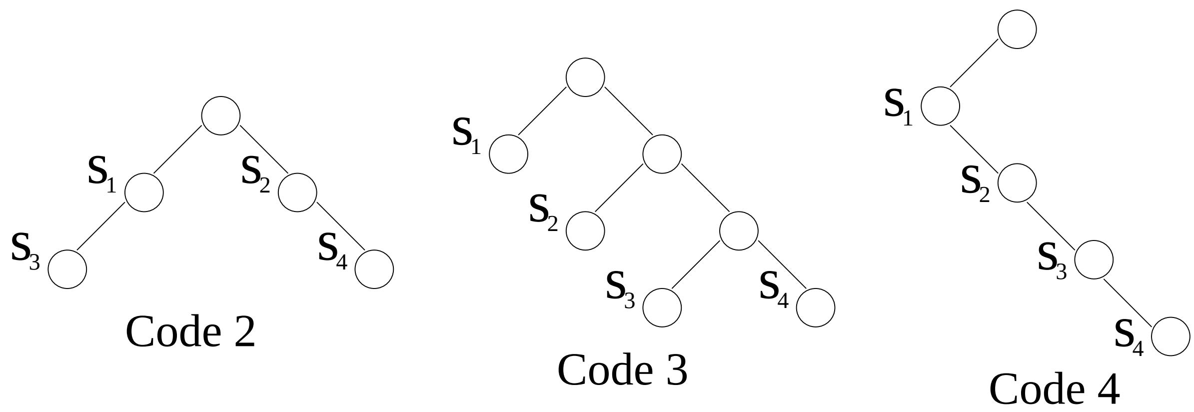
Table 1: Four different codes for a four-symbol alphabet; entropy of this source: 1.75 bits/symbol

Average Length of a Code:

$$l_{avg} = \sum_i l_i \, p_i$$

l_i: codeword length (in bits) of the codeword corresponding to symbol s_i.

Binary Trees



Code Characteristics

- *Unique decodability*
- *Prefix codes:* **no codeword is a prefix of another code-word.**
- **A prefix code is always uniquely decodable (the converse is not true)**
- **In a prefix code the code words are only associated with the** *external nodes* **of the binary tree**

HUFFMAN ENCODING (1952)

Huffman Codes are:

- Prefix codes
- Optimal for a given model (set of probabilities)

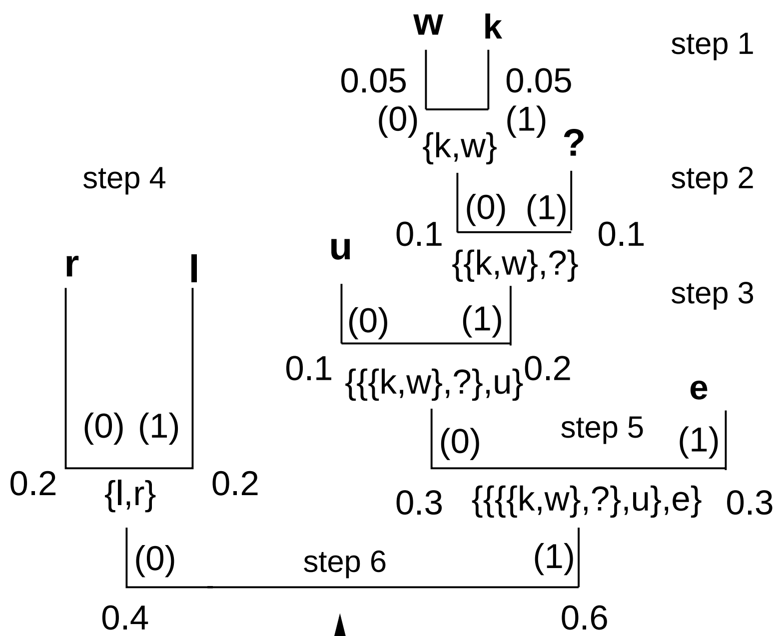
Huffman coding is based on the following two observations:

In an optimum code,

- symbols that occur more frequently will have shorter codewords than symbols that occur less frequently
- the two symbols that occur least frequently will have the same length

Example

Symbol	Probability	Codeword
k	0.05	10101
l	0.2	01
u	0.1	100
w	0.05	10100
e	0.3	11
r	0.2	00
?	0.1	1011



↑ Generate Codewords

↓ Merge Symbols

	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6
k 1/16	e 0.3	e 0.3	e 0.3	e 0.3	{l,r} 0.4	{{{{k,w},?},u},e} 0.6
l 0.2	l 0.2	l 0.2	l 0.2	{{{k,w},?},u} 0.3	e 0.3	{l,r} 0.4
u 0.1	r 0.2	r 0.2	r 0.2	l 0.2	{{{k,w},?},u} 0.3	
w 1/16	u 0.1	u 0.1	{{k,w},?} 0.2	r 0.2		
e 0.3	? 0.1	? 0.1	u 0.1			
r 0.2	k 1/16	{k,w} 1/8				
? 0.1	w 1/16					

PROPERTIES OF HUFFMAN CODES

- $H(S) \leq l_{avg} < H(S) + 1$
- **if** $p_{max} < 0.5$ **then** $l_{avg} < H(S) + p_{max}$
- **if** $p_{max} \geq 0.5$ **then** $l_{avg} < H(S) + p_{max} + 0.086$
- $H(s) = l_{avg}$ **if** $l_i = I(s_i) = -\log p_i$**, i.e., the probabilities of the symbols are of the form** 2^k
- **For a N-th extention of the DMS:** $H(S) \leq l_{avg} < H(S) + 1/N$
- **The complement of a Huffman code is also a valid Huffman code**
- **A minimum variance Huffman code is obtainied by placing the combined letter in the sorted list as high as possible**
- **Code efficiency** $= H(S)/l_{avg}$

HUFFMAN DECODING

Bit-Serial Decoding: **Fixed input bit rate but variable output symbol rate**

1. Read the input compressed stream bit by bit and traverse the tree until a leaf node is reached.

2. As each bit in the input stream is used, it is discarded. When the leaf node is reached, the Huffman decoder outputs the symbol at the leaf node. This completes the decoding for this symbol.

HUFFMAN DECODING

Lookup-Table-Based Decoding: **Variable input bit rate and constant decoding symbol rate**

Lookup-Table Construction: **If the longest codeword is L bits, then a 2^L-entry lookup table is needed.**

- **Let symbol s_i be associated with codeword c_i of length l_i. An L-bit address is formed in which the first l_i bits are c_i and the remaining $L - l_i$ bits take on all possible combinations of zero and one. Thus for the symbol s_i, there will be 2^{L-l_i} addresses.**
- **At each entry we form the two-tuple (s_i, l_i).**

Lookup-Table Decoding:

- **Read L bits into the buffer.**
- **Use the L-bit word as an address of the lookup table and obtain the corresponding symbol, say s_k, of length l_k.**
- **The first l_k bits are discarded from the buffer, and the next l_k bits are appended to the buffer from the input.**
- **Repeat 2 and 3.**

HUFFMAN CODES WITH CONSTRAINED LENGTH

Let $S = \{s_1, s_2, \cdots, s_N\}$**,** $p_1 \geq p_2 \geq \cdots \geq p_N$**, and** L **is the maximum allowed codeword length.**

- **Partition** S **into**
  $S_1 = \{s_i | p_i > 2^{-L}\}$
  $S_2 = \{s_i | p_i \leq 2^{-L}\}$
- **Create a symbol** Q **with probability**

$$q = \sum_{i \in S_2} p_i$$

- **Augment** S_1 **by Q to form** W**. Construct an optimal Huffman code for** W**. Let** c_{s1} **and** c_q **be the correpsonding codewords. Encoding:** c_{s1} **is used for** $s_i \in S_1$**;** c_q **followed by an** L**-bit fixed-length binary representation for** s_i **is used if** $s_i \in S_2$**.**

$$H(S) \leq l_{sh} \leq H(S) + 1 - q \log_2 q$$

Examples

Unconstrained length Huffman codewords

Symbol s_i	p_i	l_i	**Codeword**
0	**0.28200**	**2**	**11**
1	**0.27860**	**2**	**10**
2	**0.14190**	**3**	**011**
3	**0.13890**	**3**	**010**
4	**0.05140**	**4**	**0011**
5	**0.05130**	**4**	**0010**
6	**0.01530**	**5**	**00011**
7	**0.01530**	**5**	**00010**
8	**0.00720**	**6**	**000011**
9	**0.00680**	**6**	**000010**
10	**0.00380**	**7**	**0000011**
11	**0.00320**	**7**	**0000010**
12	**0.00190**	**7**	**0000001**
13	**0.00130**	**8**	**00000001**
14	**0.00070**	**9**	**000000001**
15	**0.00040**	**9**	**000000000**

Constrained-length (L=7 bits) Huffman codewords

Symbol s_i	p_i	l_i	Codeword	Additional
0	0.28200	2	11	
1	0.27860	2	01	
2	0.14190	3	101	
3	0.13890	3	100	
4	0.05140	4	0011	
5	0.05130	4	0010	
6	0.01530	5	00011	
7	0.01530	5	00010	
8	0.00720	11	0000	0001000
9	0.00680	11	0000	0001001
10	0.00380	11	0000	0001010
11	0.00320	11	0000	0001011
12	0.00190	11	0000	0001100
13	0.00130	11	0000	0001101
14	0.00070	11	0000	0001110
15	0.00040	11	0000	0001111

Constrained-length (L=7 bits) Huffman codewords: Ad-hoc and Voorhis methods

Symbol s_i	p_i	Unconstrained	Ad-hoc	Voorhis
0	0.28200	11	11	11
1	0.27860	10	01	10
2	0.14190	011	101	011
3	0.13890	010	100	010
4	0.05140	0011	0010	0011
5	0.05130	0010	0001	0010
6	0.01530	00011	001100	00011
7	0.01530	00010	001101	00010
8	0.00720	000011	000010	0000111
9	0.00680	000010	000011	0000110
10	0.00380	0000011	000000	0000101
11	0.00320	0000010	000001	0000100
12	0.00190	0000001	0011110	0000011
13	0.00130	00000001	0011111	0000010
14	0.00070	000000001	0011100	0000001
15	0.00040	000000000	0011101	0000000
l_{avg}		2.6940	2.7141	2.7045

HUFFMAN CODES WITH CONSTRAINED LENGTH APPLICATION

ITU-T Rec. T.4 (also known as group 3)

- **Each binary image scan line is a sequence of alternating black and white runs which are encoded separately.**
- **There are 1728 pixels in a scan line**
- **A Huffman code of 90 entries is used**
- **Huffman code table is static**
- **AN EOL codeword s used for error-detection purposes**

EXTENDED HUFFMAN CODES

Example

Symbols	Probability	Code
s_1	0.8	0
s_2	0.02	11
s_3	0.18	10

Table 2: Huffman code; entropy=0.816 bits/symbol; $l_{avg} = 1.2$ bits/symbol; redundancy=0.384 bits/symbol, or 47% of entropy

Letter	Probability	Code
s_1s_1	0.64	0
s_1s_2	0.016	10101
s_1s_3	0.144	11
s_2s_1	0.016	101000
s_2s_2	0.0004	10100101
s_2s_3	0.0036	1010011
s_3s_1	0.1440	100
s_3s_2	0.0036	10100100
s_3s_3	0.0324	1011

Table 3: The extended alphabet and corresponding Huffman code; $l_{avg} = 1.7516$ bits/new symbol; or $l_{avg} = 0.8758$ bits/original symbol; redundancy=0.06 bits/symbol, or 7% of entropy

LIMITATIONS OF HUFFMAN CODING

- To achieve the entropy of a DMS, the symbol probabilities should be negative powers of 2 (i.e., $\log p_i$ is an integer)
- Can not assign fractional codelengths
- To improve coding efficiency we can encode the symbols of an extended source. However number of entries in Huffman table grows exponentially with block size.
- Can not efficiently adapt to changing source statistics. Preset Huffman codes with different sources may result in data expansion.
- For adaptivity Huffman can be applied as a two-pass process. Works well when symbol probabilities remain constant.
- Arithmetic coding solves many limitations of Huffman coding.