

Kraft inequality

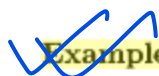
A prefix code always satisfy the Kraft inequality

$$\sum_{k=1}^L 2^{-n_k} \leq 1$$

where $n_1 \leq n_2 \leq n_3 \leq \dots \leq n_L$ are the lengths of the codewords.

Note: Kraft inequality does not tell us that a source code is a prefix code. Rather, it is merely a condition on the codeword lengths of the code and not on the code words themselves.


D.11

 **Example 2.12** Discuss the possibility of finding a prefix code with codeword lengths 1, 2, 3, 3.

Solution Since

$$K(\mathcal{C}) = \sum_{j=1}^n 2^{-l_j} = \frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \frac{1}{2^3} = 1$$

the lengths of the codewords satisfy the **Kraft inequality**, it is possible to find a prefix code with these codeword lengths (Figure 2.9(b)).



For **example**, consider the two codes in **Example 2.10**, (0, 10, 110, 1111) and (0, 10, 110, 111). The lengths of both codes satisfy the **Kraft inequality**. The lengths 1, 2, 3, 4 of the first code give

$$\frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \frac{1}{2^4} < 1$$

The lengths 1, 2, 3, 3 of the second code give

$$\frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \frac{1}{2^3} = 1$$

The **Kraft inequality** becomes equality when the code cannot be shortened.

Example 2.14 *If a code is a prefix code, what can we conclude about the lengths of the codewords?*

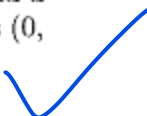
Solution Since prefix codes are uniquely decodable, they must satisfy the **Kraft inequality**.

Example 2.15 shows that, given a code with codeword lengths that satisfy the **Kraft inequality**, you cannot conclude that the code is a prefix code.

Example 2.15 *Consider code (0, 10, 110, 1011) for (A, B, C, D). This is not a prefix code as the second codeword 10 is the prefix of the last codeword 1011, despite the lengths of the codewords being 1, 2, 3, 4 which satisfy the **Kraft inequality***

$$\frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \frac{1}{2^4} < 1$$

However, since the lengths satisfy the **Kraft inequality**, we can always find a prefix code with the same codeword lengths (1, 2, 3, 4 respectively), such as (0, 10, 110, 1111).



Example 1:

The source of information A generates the symbols {A0, A1, A2, A3 and A4} with the corresponding probabilities {0.4, 0.3, 0.2 and 0.1}. Encoding the source symbols using Huffman encoder gives:

| Source Symbol | P _i | Binary Code | Huffman |
|------------------|----------------|-------------|---------|
| A0 | 0.4 | 00 | 0 |
| A1 | 0.3 | 01 | 10 |
| A2 | 0.2 | 10 | 110 |
| A3 | 0.1 | 10 | 111 |
| L _{avg} | H = 1.846 | 2 | 1.9 |

The Entropy of the source is

$$H = - \sum_{i=0}^3 P_i \log_2 P_i = 1.846 \text{ bit/symbol}$$

Since we have 4 symbols ($4=2^2$), we need 2 bits at least to represent each symbol in binary (fixed-length code). Hence the average length of the binary code is

$$L_{\text{avg}} = \sum_{i=0}^3 P_i l_i = 2 (0.4 + 0.3 + 0.2 + 0.1) = 2 \text{ bit/symbol}$$

Thus the efficiency of the binary code is

$$\eta = \frac{H}{L_{\text{avg}}} = \frac{1.846}{2} = 92.3\%$$

The average length of the Huffman code is

$$L_{avg} = \sum_{i=0}^3 P_i l_i = 0.4 * 1 + 0.3 * 2 + 0.2 * 3 + 0.1 * 3 = 1.9 \text{ bit/symbol}$$

Thus the efficiency of the Huffman code is

$$\eta = \frac{H}{L_{avg}} = \frac{1.846}{1.9} = 97.16\%$$

The Huffman encoder has the closest efficiency to the entropy that can be obtained using a prefix code. Higher efficiency can be yielded with the arithmetic coding.

Clip slide

- **Procedure to obtain Shannon-Fano code:**
- **Step 1:** list the source symbols (message) in the order of decreasing probability.
- **Step 2:** partition the set of symbols into two sets that are as close to being equiprobable as possible.
- **Step 3:** Assign 0 to each message in the upper set and 1 to each message in the lower set.
- **Step 4:** Continue this process, each time partitioning the sets with as nearly equal probabilities as possible until further partitioning is not possible.

- By following this procedure we get the code for each message as shown in table.

| Message | Probability | Column I | Column II | Column III | Column IV | Column V | Codeword | No. of bits per code word |
|----------------|-------------|-----------------------|-----------------------|-----------------------|-----------------------|------------------|----------|---------------------------|
| m ₁ | .5 | 0 <u>Partition</u> | | | | | 0 | 1 |
| m ₂ | .125 | 1 | 0 <u>Partition</u> | 0 <u>Partition</u> | | | 100 | 3 |
| m ₃ | .125 | 1 | 0 <u>Partition</u> | 1 | | | 101 | 3 |
| m ₄ | .0625 | 1 | 1 | 0 <u>Partition</u> | 0 <u>Partition</u> | | 1100 | 4 |
| m ₅ | .0625 | 1 | 1 | 0 <u>Partition</u> | 1 | | 1101 | 4 |
| m ₆ | .0625 | 1 | 1 | 1 | 0 <u>Partition</u> | | 1110 | 4 |
| m ₇ | .03125 | 1 | 1 | 1 | 1 | <u>Partition</u> | 11110 | 5 |
| m ₈ | .03125 | 1 | 1 | 1 | 1 | 1 | 11111 | 5 |