# JPEG Image Compression Algorithm and the DCT

## Contents



## 1 Entropy Coding

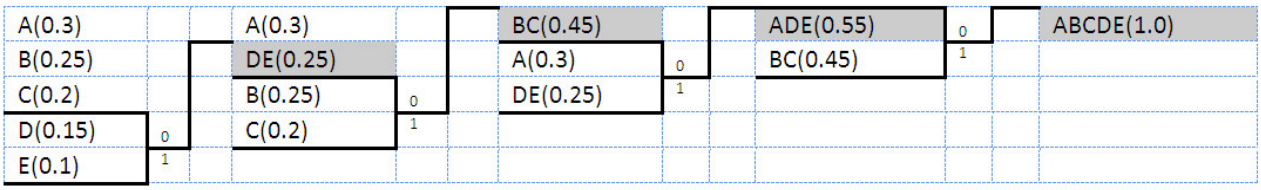
A lossless coding system attempts to reduce the redundancy inherent in the unequal probabilities of some numerical values compared with others. This is typically done by allocating binary codes of different lengths so that values with a high probability are associated with short code-words and the long code-words are reserved for the low-probability values (a similar principle is the basis of Morse code). Entropy coding reduces the average number of binary digits to be sent per numerical value (relative to PCM). This improved efficiency, however, is obtained at a cost of much greater complexity.

### 1.1 Huffman Coding

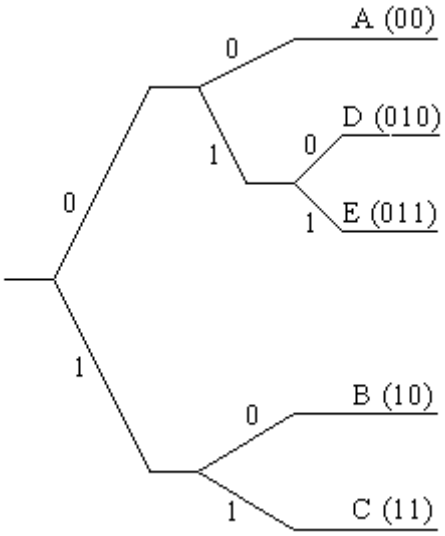
¡p¿There are a number of standard ways of carrying out such;*entropy coding*, of which the most well-known is probably *Huffman coding*. Assuming the numerical values to be coded have been converted into a set of "symbols," the Huffman algorithm for assigning binary codes to these symbols is:

1. list the symbols in descending order of probability
2. combine the bottom two symbols into a new compound symbol whose probability is the sum of the two
3. repeat steps 1 and 2 until you have a single compound symbol whose probability is equal to 1
4. generate a decision tree which reverses the process and code each decision by a 1/0
5. read the code for each symbol off the decision tree

As an example, consider a situation involving five symbols $\{A, B, C, D, E\}$ with probabilities $\{0.3, 0.25, 0.2, 0.15, 0.1\}$:




The resulting decision tree is:




**Q**: Decode the following: $1001100010011010$ and calculate the average number of bits per symbol used in this small example.
The example code developed above would need *on average* 2.25 binary digits per "symbol" (for long enough input sequences) whereas a PCM-like approach would have required a *constant* 3 digits per symbol (5 is more than 4 but not more than 8), so the benefit of doing this coding is that you would need to send 25% fewer binary digits (on average).

## 1.2 Problem with Huffman Coding

A problem with Huffman coding is that, for it to provide an optimal scheme, it requires all symbols to have probabilities that are equal to $p = 1/2^{l(c)}$ where $l(c)$ is the length of the code word for symbol $c$. This can be understood from the following argument.
Average self information or Entropy for a sequence of symbols being communicated is given by:

$$H = -\sum p(c) \log_2(p(c))$$

where $p(c)$ is the probability of symbol $c$ drawn from alphabet $C$. Each symbol is represented with a code word of length $l(c)$. We can then see that the average code word length is given by:

$$L = \sum p(c) l(c).$$

Code words in Huffman coding are represented in binary with 1 or more binary values. This means $l(c) = 1$, so that

$$L = H.$$

A Huffman code for a sequence of symbols being communicated will be optimal if the sequence being communicated can be as short as possible. Therefore the best possible representation is given by $L = H$.
We can observe that this occurs when the length of a code word is equal to the self information for that symbol, i.e. $l(c) = -\log_2(p(c))$. Thus for optimal Entropy coding using Huffman coding the probabilities for each of the symbols must follow:

$$p(c) = 1/2^{l(c)}.$$

## 1.3 Alternative Entropy Coding Technique: Arithmetic Coding

Huffman coding is used very much in image compression. Another Entropy coding technique is Arithmetic coding. Arithmetic coding can provide optimal coding even if the probabilities of the different symbols do not follow any particular rule. There are numerous different variations of arithmetic coding. A form of arithmetic coding is used in video compression including the latest video compression standard H.265.