IBM Technical Disclosure Bulletin Vol. 22 No. 5 October 1979

# PADDING OF HUFFMAN STRINGS TO CONVENIENT BOUNDARIES

D. A. Hamilton, P. R. Herrold and M. J. Ossefort

Disclosed is a technique to enhance storage efficiency for Huffman encoded messages by avoiding the use of either a special terminator or length prefix to mark the end of the message.

Huffman encoding is a method of assigning variable-length bit strings to represent the characters in an alphabet such that frequent characters receive short representations while infrequent characters are assigned longer representations. The result is that, overall, some redundancy is removed and the storage requirement of text can be reduced. Unfortunately, there can be no certainty that the encoded "message" will end on a convenient byte (or any other) boundary. To mark the end of a message, past solutions have used "length codes" or special "terminator" characters.

The disadvantage of either of these two popular methods is that both require that some "extra" bits be added to each message. These additional bits may well force the message to spill over into an additional byte.

Often, one can know "for free" that the message (before any length code or terminator is added) fits into some number of bytes with zero to seven bits left over. An example of this situation is a dictionary in which entries are grouped into separate arrays, depending on the minimum number of bytes required for a particular message. In situations such as this, the possibility that a length code or terminator could force the message length over the next byte boundary is quite significant.

Unless the alphabet is very small and the relative frequencies very even, there will be at least one character that will be encoded with a bit string longer than 8 bits. This happens if there is any character that occurs less frequently than 1 out of 256. This character can be used as "padding" if a rule is made that any "unfinished" character at the end of a message is thrown away. For example, if a message is 3 bits short of filling out the last byte, the first 3 bits of the long character are used as padding. (Since we know this is the last byte, we stop at the byte boundary and discard the incomplete character at the end.)

To make things even simpler, one can force the long character to be represented by all zeroes. This can be done by exclusive-ORing the representation assigned by the Huffman algorithm to the infrequent character with all the assignments (including itself). When exclusive-ORing with a shorter string, only as many bits of the long string as

needed are used. The reason this works is that Huffman codes can be viewed as specifying a "path" through a binary tree: a "1" means "go right", a "0" means "go left". The exclusive-OR operation swaps right and left subtrees of any node, as needed, so that the longest character is on an all-zero path.