Guna sai kiran
BT20ECE075
LAB-01

I. Aim:

Huff-Mann Encoding and Decoding Using Matlab.

II. Software Required:

Matlab;

III. Theory:

Huffman coding is an efficient source coding algorithm for source symbols that are not equally probable. A variable length encoding algorithm was suggested by Huffman in 1952, based on the source symbol probabilities P(xi), i=1,2. . . . . ,L

Huffman coding is a lossless data compression algorithm. The main principle in Huffmans coding is that each message in encoded using individual or combination of 0 or 1 bits. Here the most frequent characters will be assigned with the smallest code and the least frequent character will have the largest code. Hence Huffman's coding is based on the variable length approach. Also in Huffman's there is no repetition of the code for the same message.

Data compression has become a necessity while processing information that occupies huge memory. Data compression not only reduces the size but also helps to increase the processing time. Huffman's coding is based on the frequency of occurrence of a data item. The principle is to use a lower number of bits to encode the data that occurs more frequently. Codes are stored in a code book. In all cases the code book plus encoded data must be transmitted to enable decoding.

The algorithm is optimal in the sense that the average number of bits required to represent the source symbols is a minimumprovided the prefix condition is met.

IV. Steps:

The steps of Huffman coding algorithm are given below:

- Arrange the source symbols in increasing order of heir probabilities.
- Take the bottom two symbols tie them together .Add the probabilities of the two symbols write it on the combined node. Label the two branches with a "1 a "0"
- Treat this sum of probabilities as a new probability associated with a new symbol. Again pick the two smallest probabilities, tie them together to form a new probability. Each time we perform the combination of two symbols we reduce the total number of symbols by one. Whenever we tie together two probabilities (nodes) we label the two branches with a "0 a "1 .
- Continue the procedure until only one procedure is left (it should be one if your addition is correct). This completes the construction of the Huffman Tree.

- To find out the prefix codeword for any symbol, follow the branches from the final node back to the symbol. While tracing back the route read out the labels on the branches. This is the codeword for the symbol.

V. Inbuilt Functions:

1) DICT :**HUFFMANDICT(SYM, PROB):**
Generates a binary Huffman code dictionary usingthe maximum variance algorithm for the distinct symbols given by the SYM vector. The symbolscan be represented as a numeric vector or single-dimensional alphanumeric cell array. Thesecond input, PROB, represents the probability of occurrence for each of these symbols. SYMand PROB must be of same length

2) Huffmanenco. This function is used in Huffman coding. The syntax is:**huffmanenco(sig,dict):** This line encodes the signal 'sig' described by the 'dict' dictionary. The argument 'sig' can have the form of a numeric vector, numeric cell array or alphanumeric cell array. If 'sig' is a cell array, it must be either a row or a column. The 'dict' is an Nx2 cell array, where 'N' is the number of distinct possible symbols to be encoded. The first column of 'dict' represents the distinct symbols and the second column represents the corresponding codewords. Each codeword is represented as a numeric row vector, and no codeword in 'dict' can be the prefix of any other codeword in 'dict'. You can generate 'dict' using the huffmandict function.

3) Huffmandeco. This function is used in Huffman decoding. The syntax is:
huffmandeco(comp,dict): This line decodes the numeric Huffman code vector comp using the code dictionary 'dict'. The argument 'dict' is an Nx2 cell array, where 'N' is the number of distinct possible symbols in the original signal that was encoded as 'comp'. The first column of 'dict' represents the distinct symbols and the second column represents the corresponding codewords. Each codeword is represented as a numeric row vector, and no codeword in 'dict' is allowed to be the prefix of any other codeword in 'dict'. You can generate 'dict' using the Huffmandict function and 'comp' using the huffmanenco function. If all signal values in 'dict' are numeric, 'dsig' is a vector; if any signal value in 'dict' is alphabetical, 'dsig' is a one-dimensional cell array.

VI. Conclusion:

We have successfully Encoded and Decoded Of huff-mann Coding Using Matlab.