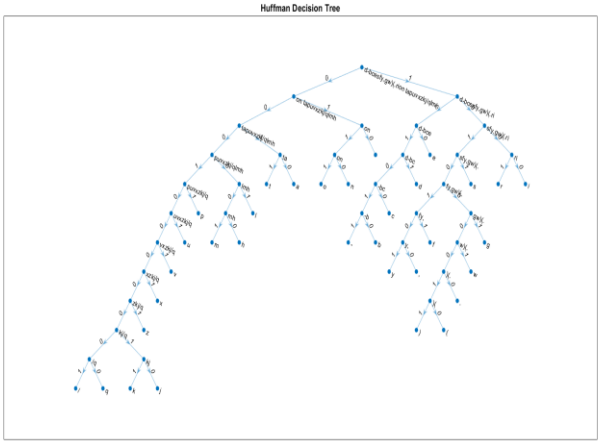


PROJECT 1 REPORT

Huffman Algorithm

ABSTRACT

Transmitting the data has faced many major challenges since the beginning of the computer science era. One of the biggest challenges is to compress the data efficiently without losing information. The optimum solution is Huffman algorithm, which will be addressed here.

Ahmed Wael

Information Theory and Coding CIE425

Table of Contents

Introduction

The field of computer science has been evolving since the 1950s, especially in the subfield of information theory. This subfield was solely established by Claude Shannon when he published his evolutionary paper "A mathematical theory of communication" [1]. Shannon defined a new mathematical quantity known as the Entropy in order to quantify the smallest number of bits needed to represent a given source symbol, or in other words the average information content [1]. Collaborating with his colleague Fano, they invented Shannon-Fano code in order to reach the Entropy [2]. While this code achieves high efficiency, it is not the optimum lossless source code. Instead, the optimum source code was invented by Shannon's PhD student David A. Huffman as he proved this optimality in his paper "A Method for the Construction of Minimum-Redundancy Codes" [3].

Theory

The mathematical formula for the Entropy is $E = -\sum_{<k>} p_k * \log_2(p_k)$, where p_k is the probability for each symbol 'k' in the alphabet. This was derived from the definition of the information which is $I(s_k) = log_2(\frac{1}{p_k})$, and by definition, the entropy is the average information content.

ALGORITHM

1. Convert the stream of symbol into a stream of decimal numbers
    a. Get the ascii code for each symbol

2. Calculate the entropy using Shannon formula.

3. Evaluate the run-length code and calculate its efficiency.

4. Construct the decision tree
    a. Sort the probabilities of the alphabet symbols in a descending order.
    b. Sum the least two probabilities.
    c. Make a binary tree, with the parent is equal to the sum of the least two probabilities, and the children equal to the two least probabilities.
    d. Assign weight = 1 for the edge connecting the parent node and the smallest number node, and weight = 0 for the edge connecting the parent node and the second smallest node.
    e. Repeat the steps from 1 to 4 until there is only one symbol with probability = 1.

5. Giving weights for each alphabet symbol
    a. For each symbol in the alphabet do the following
        i. Traverse the tree from the root using the binary search algorithm, where the half of the tree is ignored each iteration, and concatenate the weights in each step.

6. Encode the stream of symbols
    a. Loop through the stream of symbol and map each symbol to its corresponding code.

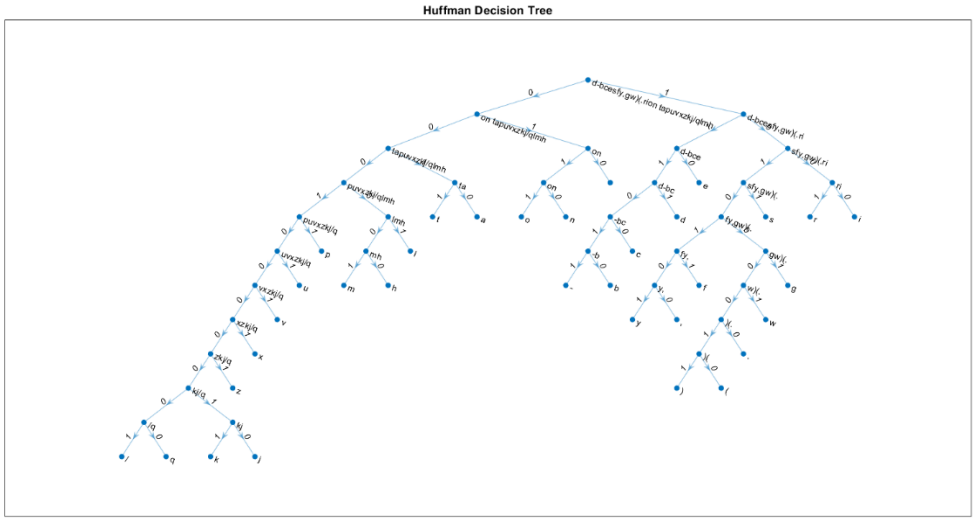
7. Decode the encoded symbols
    a. Reverse the process of encoding by looping through the encoded stream of bits and comparing it with each code and map it back into the corresponding decimal value and therefore its corresponding symbol.
8. Calculate the efficiency of the Huffman code and compare it with the run-length code efficiency.

Code details and results

Can be found in the other file named 'Code documentation and algorithms'.

Discussion

The obtained tree is given below, where it is obvious that the most probable symbols such as 'e' with a probability = 0.1040 is assigned only 3 bits, while the least probable symbols such as 'k' with probability = $7.1225*10^{-4}$ is given 11 bits. Therefore, the average length approaches the entropy more than the run-length code where each symbol is assigned 11 bits.



The efficiency of run-length code is 70.9% while the efficiency of Huffman code is 99.547%, which confirms the optimality of Huffman code.

The encoded file length using Huffman code is 6004 while the encoded file using the run-length code is 15444. This is only 38.8% of the original file size, which also indicates the efficiency of Huffman code.

References

[1] Shannon, C. (1948). A Mathematical Theory of Communication. Bell System Technical Journal, 27(4), pp.623-656.

[2] The Shannon-Fano Algorithm", *Users.cs.cf.ac.uk*, 2018. [Online]. Available: https://users.cs.cf.ac.uk/Dave.Marshall/Multimedia/node209.html. [Accessed: 03- Nov- 2018].

[3] D. Huffman, "A Method for the Construction of Minimum-Redundancy Codes", *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098-1101, 1952.