PRANSH GUPTA
E20CSE165 [EB-10]

A Critical Review of Image Compression technology based on Huffman coding

Compression is a method for minimising the total data size while keeping the clarity of the multimedia content. Image compression is a viable approach since the distribution and storage of reduced image files is significantly faster and more efficient than that of raw uncompressed data. For media data compression, there are numerous techniques and formats available, such as Arithmetic coding, Huffman encoding, and so on. This review focuses on Huffman coding.

One of the core Greedy lossless data compression algorithms is Huffman encoding. It implements variable length encoding, which generates a variable length code for every letter dependent on how frequently it repeats in the provided data. The character with the maximum frequency of occurrence is assigned the smallest code, while the character with the minimum frequency of occurrence is assigned the largest code. All the while, the prefix rule is used to avoid ambiguity when decoding.

<u>Working</u> –

The approach for Huffman Encoding is based on a binary-tree frequency-sorting method, which begins by constructing a binary tree called the Huffman tree, each node of which is represented by a byte symbol and the frequency of that byte on the input. Steps of construction :-

- Scan the data to find out the frequency of occurrence of each byte/letter.
- Then the nodes are put in a reverse priority queue depending on their frequencies (lowest frequency gets highest priority).
- After which we start a loop, which continues until the queue is empty.
- We proceed by removing two nodes from the queue and adding them to form an internal node which will resultantly have a frequency equal to the sum of the two nodes'.
- Then the newly generated internal node is added to the queue and the above step repeats.
- In the end, the root of the tree is the final node on the queue.

PRANSH GUPTA
E20CSE165 [EB-10]

Time Complexity Analysis –

ExtractMin() is executed **2*(n-1)** times when there are **n** nodes. Furthermore, because extractMin() invokes **minHeapify**, it takes **O(logn)** runtime . As a consequence, Huffman Coding has a total time complexity of **O(nlogn)** [where n is the total no. of different letters in the given data].

Advantages –

- The Huffman encoding strategy makes use of the difference in frequency and utilises less data/storage for frequently occurring characters while requiring more data/storage for each of the more infrequent characters.
- It is also simple to implement and takes less time to execute than many other lossless techniques, such as Arithmetic coding.
- The binary codes generated are prefix-free.

Disadvantages –

- The Huffman code bears the constraint of only being able to assign integer length codewords. This frequently leads in suboptimal performance especially in situations of fractional information content. For instance, 4.23 bits are represented by 5 bits codewords.
- When opposed to lossy encoding approaches, lossless data encoding schemes, such as Huffman encoding, yields a lower compression ratio. Even the arithmetic algorithm outperforms it in terms of compression ratio.

My Opinion :

Acc. to me it depends upon the users requirements and need. If they want an easy to use algorithm with fairly accurate solution and don’t have much problem with its integer length codewords they can very well go for it but if we are talking about highly advance computations with storage constraints they should try searching for a better alternative.