HUFFMAN CODING

Name: RUSHIL PRAJAPATI (21BCE232) ,

RAJ PARMAR (21BCE240),

FENIL RAMOLIYA (21BCE244),

Subject Name : DIGITAL COMMUNICATION INNOVATIVE ASSIGNMENT :

HUFFMAN ENCODING TECHNIQU E

NOVEMBER 1, 2022

ABSTRACT

Huffman coding is an approach to text compression developed by David A. Huffman during his Ph.D. An MIT student, published in 1952 in his thesis “Method for the Construction of Minimally Redundant Code". In computer science and information theory, this is his one of many lossless data compression algorithms. This is his statistical compression technique that converts characters to his bit strings of variable length and generates a prefix code. The most frequently occurring character is converted to the shortest bit string.

ABOUT DATA COMPRESSION

Data reduction or Data compression is a data preprocessing technique that can be applied to obtain a reduced or compressed representation of a dataset. In other words, mining on a reduced dataset should be more efficient while providing the same analytical results. Data compression is useful when using cryptographic mechanisms to reduce the size of data sets. Data compression applies data encoding or transformations to obtain a reduced or compressed representation of the original data. Huffman encoding is a successful compression technique originally used for text compression. Huffman's idea was to represent the common characters in the source with short codewords, instead of using fixed length codes like 8-bit extended ASCII for each symbol, and the less common characters It is represented by a long codeword.

INTRODUCTION

Suppose you need to store a string of length 1000 containing characters p, r, e, l. Storing it as a single-byte character requires 1000 bytes (or 8000 bits) of memory. If the symbols in the string are encoded as (00=p, 01=r, 10=e, 11=l) then 1000 symbols can be stored in 2000 bits, saving 6000 bits of storage.

The number of times a symbol occurs in a string is called frequency. If the frequencies of different symbols within a string vary significantly, the symbols can be assigned variable-length codes based on their relative frequencies. The most common characters can be represented using shorter codes than those used for less common source symbols. The greater the variation in the relative frequency of symbols, the more advantageous it is to reduce the size of strings encoded using variable-length codes.

Huffman was able to design the most efficient compression method of this type, no further mapping of individual resource symbols to unique strings of bits will require less space to store the part text when the actual frequencies of the symbols agree with the frequencies used to create them code.

HOW IT WORKS IN PROGRAMMING POV

In Huffman coding, the complete code set can be represented as a binary tree which is also known as the huffman tree. In terms of programming language tree is one of the data structures which has e parent node and that parent node contains n number of children nodes, ahead of that each children node acts as sub-parent node and may contain m number of children which are grandchildren of our parent node. Binary tree is one type of tree which has a special property, every parent node can only contain at most 2 children nodes. Either it can be 0,1 or 2 children. Here, huffman coding uses a tree named complete binary tree which means all root/parent nodes must contain 2 children except terminal nodes. Here, for our convenience and traversing we provide bit ‘0’ which represents the following left child and bit ‘1’ represents the following right child. One code bit represents each level so that the characters which are more frequent are near the root/parent node and they are coded with few bits and rare characters are far from the root/parent node and they are coded with many bits. In binary tree a variant of tree called **minHeap** Tree which has a very special property, value of the root node will always be lesser than the value of the child node.

First, the original characters are stored as leaf nodes in a regular array whose size depends on the number of characters n along with the frequency of occurrence. The finished tree has a maximum of n leaf nodes and n - 1 inner node.

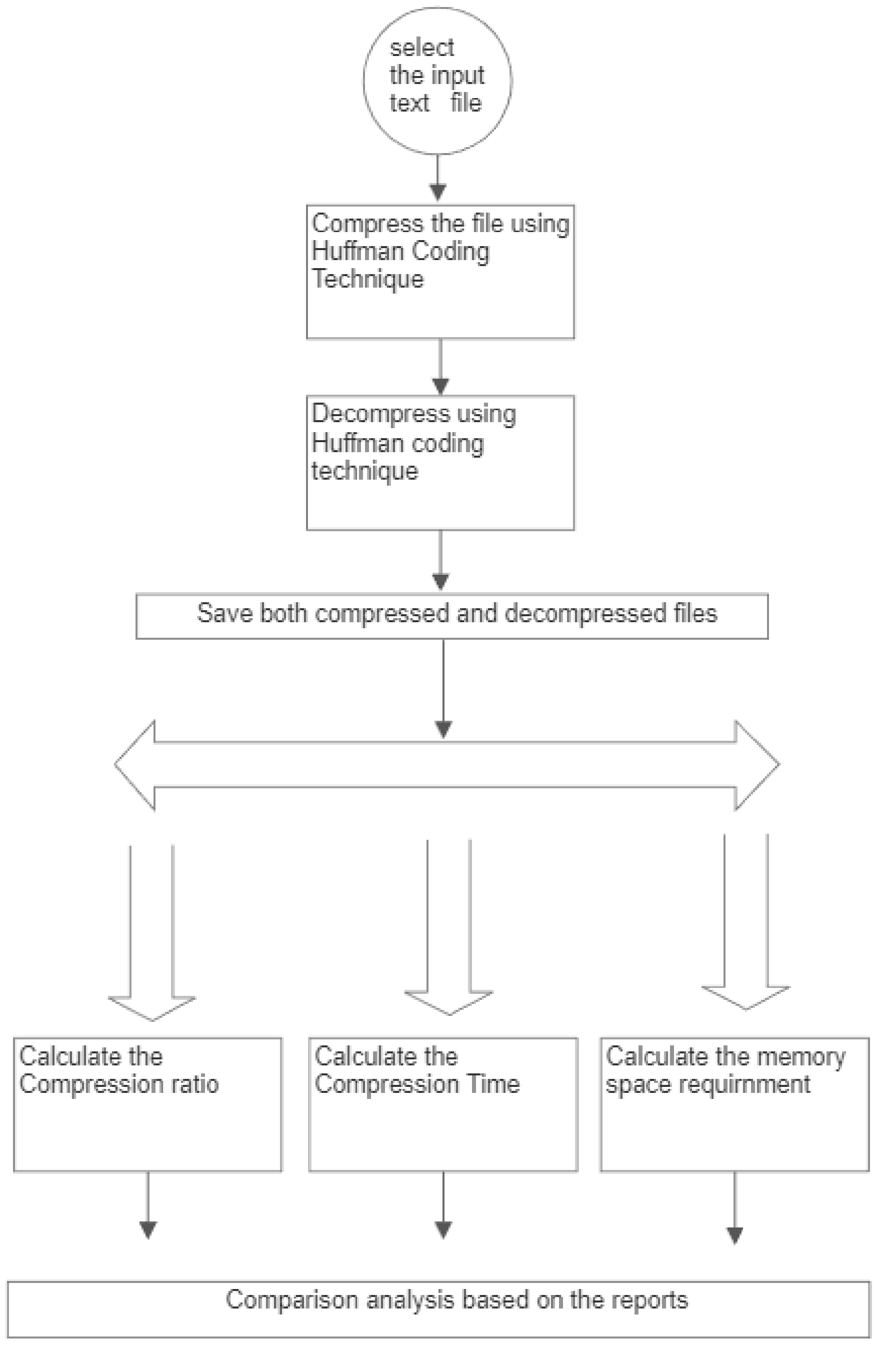
We created this entire code in C++ language using its STL and Object Oriented Properties.

` WHAT IS GIVEN AND WHAT TO FIND

Given : A set of symbols and their weights or occurrence frequencies

Find : A prefix-free binary tree with minimal expected codeword length or a tree with minimal weighted path length from the root node.

REAL TIME RESEARCH METHODOLOGY


select the input text file
Compress the file using Huffman Coding Technique
Decompress using Huffman coding technique
Save both compressed and decompressed files
Calculate the Compression ratio
Calculate the Compression Time
Calculate the memory space requirnment
Comparison analysis based on the reports

ALGORITHM FOR HUFFMAN ENCODING

Step 1: Create a node for each unique character and it’s frequency and then by using all that nodes build a min heap tree

Step 2: Extract two nodes which has least frequencies from our min Heap

Step 3: Create a new internal node which has frequency equal to the sum of the two node’s frequencies (That extracted two nodes) and then make the first extracted node as that new node’s left child and the other extracted node as its right child. After that add this node to min Heap.

Step 4: Repeat Step 2 and Step 3 until the heap tree contains only one node.

[the remaining node is the root node]

WHAT IS HUFFMAN TREE

Every Node has the frequency equal to the sum of its both child node frequencies which provide its notable property.

TRAVERSING AND HOW BITS ARE GENERATED IN OUTPUT

We provide each left child bit ‘0’ and ‘1’ to each right child, then we start travesing from the root node and reach to our desired node (charcter) and print all the bits which we seen on route towards that node.

VARIATION OF HUFFMAN CODING

a) n-ary Huffman coding

The n-ary Huffman algorithm uses the $\{0, 1, \ldots, n - 1\}$ alphabet to encode messages and builds an n-ary tree.

b) Huffman template algorithm

The Huffman Template Algorithm allows all kinds of weights (costs, frequencies, weight pairs, non-numeric weights) and one of many combination methods (not just addition).

c) Adaptive Huffman coding

Dynamically compute the probability based on the actual count of the last of the source string. This is somewhat related to his LZ family of algorithms.

d) Optimal alphabetic binary trees

The alphabetic version requires the same alphabetical order for inputs and outputs. This is because the author of the paper presented his first linear arithmetic solution to this optimal binary alphabet problem, and although it has some similarities to Huffman's algorithm, is a variation of that algorithm There is none. These optimal his alphabet binary trees are often used as binary search trees.

APPLICATION

This algorithm can be viewed as a generalization of Huffman coding. In practice, arithmetic coding is often preceded by he Huffman coding because it is easier to find the arithmetic code for his binary input than for non-binary input. Although arithmetic coding offers better compression performance than Huffman coding, Huffman coding is still widely used due to its simplicity, speed, and lack of patent burden. The Huffman encoding is now commonly used as a "backend" for other compression schemes.

REFERENCES

- Mamta Sharma : 'Compression Using Huffman Coding'. International Journal of ComputerScience and Network Security, VOL.10, No.5.
- Guy E. Blelloch : 'Introduction to Data Compression'. Carnegie Mellon University.
- https://en.wikipedia.org/wiki/Huffman_coding
- https://www.ijcstjournal.org