HUFFMAN COMPRESSION & DECOMPRESSION

Name: Hussen Adel Mohamed
ID: 5783
Name: Faris Waleed Gaber
ID:5559
Name: omar Sami Dorgham
ID:5689
Name: Ziad Hassan ElShabasi
ID:5876

1) Introduction

- In computer science and information theory, a Huffman code is a particular type of optimal prefix code that is commonly used for lossless data compression. The process of finding or using such a code proceeds by means of Huffman coding, an algorithm developed by David A. Huffman while he was a Sc.D. student at MIT, and published in the 1952 paper "A Method for the Construction of Minimum-Redundancy Codes". The output from Huffman's algorithm can be viewed as a variable-length code table for encoding a source symbol (such as a character in a file). The algorithm derives this table from the estimated probability or frequency of occurrence (weight) for each possible value of the source symbol. As in other entropy encoding methods, more common symbols are generally represented using fewer bits than less common symbols. Huffman's method can be efficiently implemented, finding a code in time linear to the number of input weights if these weights are sorted. However , although optimal among methods encoding symbols

2) Code description

a) Main idea

- ✓ **COMP Idea**
- ✓ Reading file of data
- ✓ Calculate Frequency of each character
- ✓ Create Huffman tree
- ✓ Replace every character with new code
- ✓ Get result in binary string
- ✓ Check length of string to be divisible by 8 if not adding zeros to left of string(padding)
- ✓ Write in file with string length of padding with string of binary and also flag for folder or file
- ✓ During writing collect every 8 binary character by one char character to dec. comp file
- ✓ **DECOMP Idea**
- ✓ Reading comp file and store result in string of binary
- ✓ Removing bits which contain padding and folder flag and so on
- ✓ Take binary value of each character in un-ordered map
- ✓ And rest of string is main text but in binary
- ✓ By using un-ordered map replace every binary code with equivalent char
- ✓ Write file with this new string of data

b) Data structures

- ✓ Collection of data as int ,char,float
- ✓ Struct data
- ✓ **library in c++**
- ✓ string
- ✓ priority_queue
- ✓ unorder map

c) Algorithm used

- ✓ Huffman is used which implement greedy implement
- ✓ Main idea is depend of frequency of each char
- ✓ Then creating Huffman tree by priority queue depending on frequency
- ✓ Extract min 2 times and push sum to priority then looping

d) Complexity time for Huffman

- ✓ in compress function there are many loop to do special function by O(n)
- ✓ Using a heap to store the weight of each tree, each iteration requires O(logn) time to determine the cheapest weight and insert the new weight. There are O(n) iterations, one for each item
- ✓ The time complexity of the Huffman algorithm is O(nlogn).

e) Header format of the compressed file

Byte index	1	2	3	4	5	6,7,8	9th Byte to end of file
Mention char	N	F	P	n	B	Encode length	Prefix code + encode text

Mention char	mission
N	Number of files in folder
F	Folder flag
P	padding
n	Number of used character
B	Total bytes of all char
Prefix code	Code of each character separately
Encoded File	Code of whole text

Where Prefix code : [Character][length of code][encoded code] written in this form