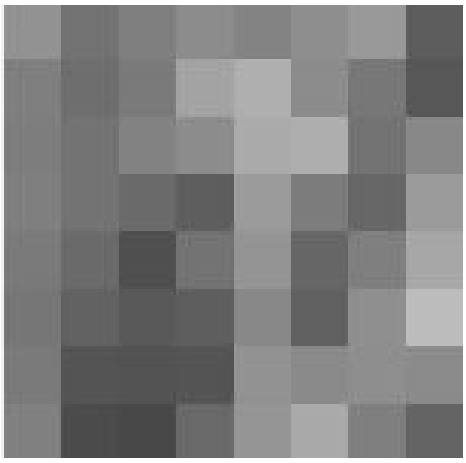
Design and Analysis of Algorithms project

“Image Compression using Huffman coding technique”



Bharani Ujjaini Kempaiah

PES1201700005

Bhavya Charan

PES1201700032

INDEX

1	ABSTRACT	2
2	INTRODUCTION	2
3	DATA STRUCTURES USED	2-3
4	SNAPSHOTS	4-5
5	RESULTS	6
6	CONCLUSION	6
7	REFERENCES	6

ABSTRACT

The project aims at obtaining a compressed form of the given image by employing the Huffman coding technique. Given a gray scale image we take as source symbols the intensity values for the pixels and construct a huffman tree using which we assign and obtain the huffman codes.

INTRODUCTION

Image compression is a type of data compression applied to digital images, to reduce their cost for storage or transmission. Algorithms may take advantage of visual perception and the statistical properties of image data to provide superior results compared with generic data compression methods which are used for other digital data.

Huffman coding is one of the basic compression methods, that have proven useful in image and video compression standards. When applying Huffman encoding technique on an Image, the source symbols can be either pixel intensities of the Image, or the output of an intensity mapping function.

DESCRIPTIONS OF DATA STRUCTURES INVOLVED

Given below are the major information storing variables and details about their functionality

1. **Image[height][width]** : This is the 2-dimensional array consisting of the pixel intensity values in the input image/matrix.
2. **Int hist[256]** : This array consists of the frequency of the corresponding pixel intensity i.e. hist[10] will contain the count of how many times the intensity value of 10 occurs in the input matrix/image. 0 is filled in by default.

3. **Struct pixfreq** : Structure which makes up the Huffman tree.

   The following are its members

   - int pix : holds the intensity value
   - float freq : holds the frequency of occurrence for corresponding intensity in pix
   - struct pixfreq *left, *right : pointers to the nodes left and right children which initially are NULL
   - char code[maxcodelen] : character array to hold the huffman code

Array pix_freq is an array of structures of pixfreq. In this array, the nodes are always added to the end. It is not sorted everytime. It only contains the nodes for those pixels that have a non-zero frequency.

4. **Struct huffcode** : structure which is used to process the active nodes that constitute the huffman tree at any given point of time.

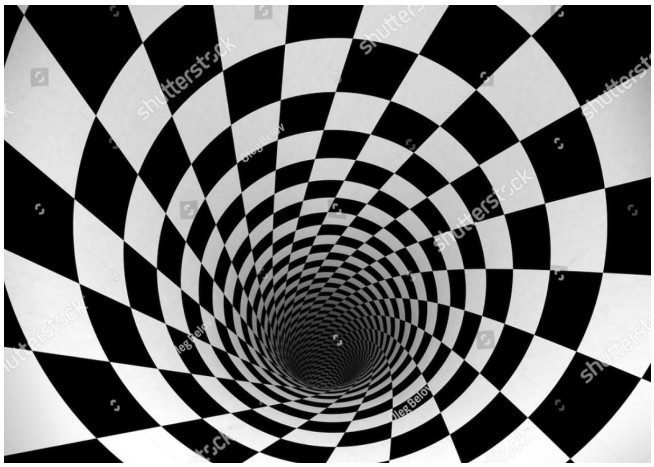
   The following are its members

   - int pix : holds the intensity value
   - int arrloc : holds the corresponding index from the pixfreq structure array
   - float freq : holds the frequency of occurrence for corresponding intensity in pix

Array huffcodes is an array of structures of huffcode. It contains only those nodes that have a non-zero frequency. This array is always sorted at any instance of time, in decreasing order of freq. New nodes are inserted at the correct position by shifting all the other nodes that are less than the current freq value by one to the right.

SNAPSHOTS

The following image was used as input for compression.



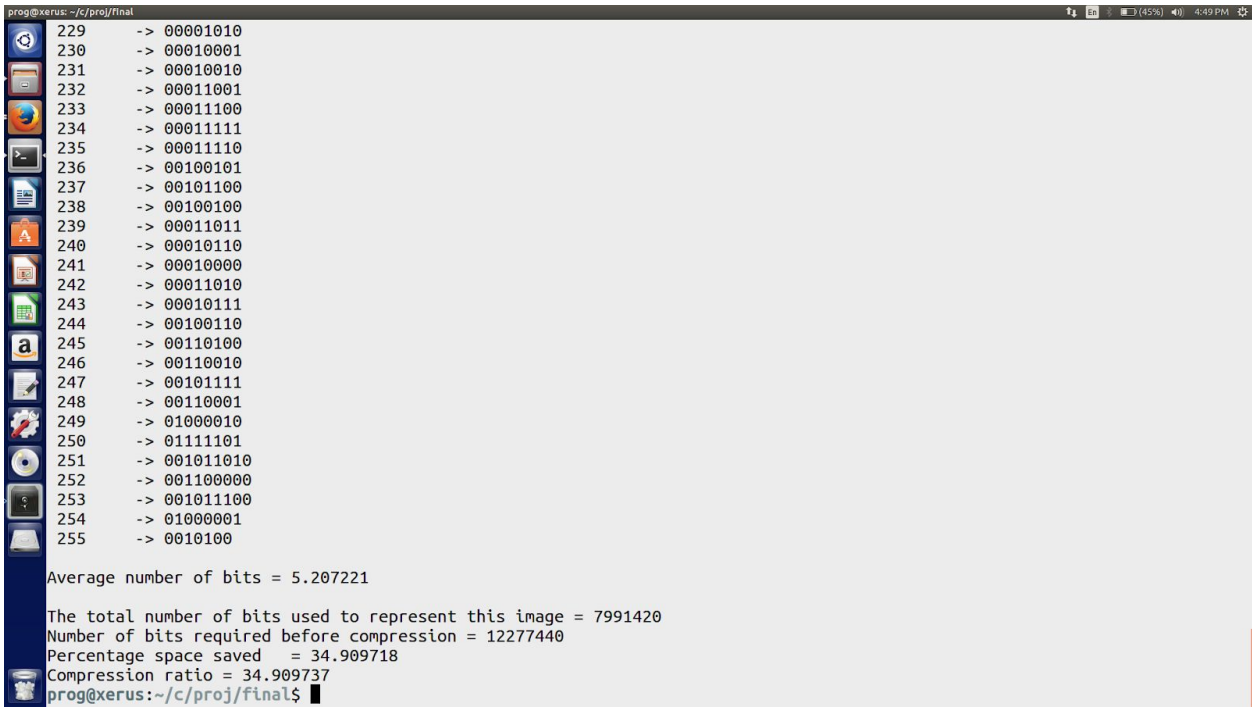
The codes obtained for each intensity value after compression is given below -

```
prog@xerus:~/c/proj/final$ ./huffman

Huffmann Codes::

pixel values -> Code

number of nodes = 256
 0       -> 1
 1       -> 01110
 2       -> 010011
 3       -> 0011110
 4       -> 00111001
 5       -> 001000010
 6       -> 0000101101
 7       -> 0011111011
 8       -> 0101011111
 9       -> 00000100001
         10      -> 00001111000
         11      -> 00011000001
         12      -> 00011000100
         13      -> 00010011111
         14      -> 00011000010
         15      -> 00010011110
         16      -> 00011101111
         17      -> 00011101100
         18      -> 00100111110
         19      -> 00101101110
         20      -> 00100111101
         21      -> 00100001111
         22      -> 00101110101
         23      -> 00110011001
         24      -> 00110011000
         25      -> 00110011011
         26      -> 00111010111
         27      -> 00111011101
```


prog@xerus: ~/c/proj/final
229 -> 00001010
230 -> 00010001
231 -> 00010010
232 -> 00011001
233 -> 00011100
234 -> 00011111
235 -> 00011110
236 -> 00100101
237 -> 00101100
238 -> 00100100
239 -> 00011011
240 -> 00010110
241 -> 00010000
242 -> 00011010
243 -> 00010111
244 -> 00100110
245 -> 00110100
246 -> 00110010
247 -> 00101111
248 -> 00110001
249 -> 01000010
250 -> 01111101
251 -> 001011010
252 -> 001100000
253 -> 001011100
254 -> 01000001
255 -> 0010100
Average number of bits = 5.207221
The total number of bits used to represent this image = 7991420
Number of bits required before compression = 12277440
Percentage space saved = 34.909718
Compression ratio = 34.909737
prog@xerus:~/c/proj/final$

RESULTS

Without using image compression, we would have required 8 bits to represent a pixel with upto 256 different values of intensities. But now using the huffman coding technique, we are able to uniquely represent each intensity value using only an average of around 5 bits. Thus from requiring as large as 1,22,77,440 bits we have reduced to 79,91,420 bits for the corresponding image. This saves around 35% space and results in a compression ratio of 34.91%.

CONCLUSION

In this project we have used Huffman Coding Technique based on minimal length coding to uniquely identify every intensity value. Using this encoded data for representing the image instead of the original helps reduce the space required to store data. The reduction in file size allows more images to be stored in a given amount of disk or memory space. It also reduces the time required for images to be sent over the Internet or downloaded from Web pages.

REFERENCES

1. Wikipedia
2. Stackoverflow
3. Introduction to the Design and Analysis of Algorithms, Anany Levitin