JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY

jiit

विद्या तत्व ज्योतिसमः

ALGORITHM AND PROBLEM SOLVING LAB

PROJECT REPORT

IMAGE COMPRESSION USING HUFFMAN CODING

SUBMITTED BY:

Utkrisht Sharma – 16103116

Aarya Patel – 16103026

Chaitanya Kumar – 16103244

BATCH – B15

Problem Statement

Compressing 24bit .bmp image using Huffman Coding.

Relevance

The technique is best suited for lossless transfer of images.

Introduction

Huffman coding is a lossless data compression algorithm. Using this algorithm we can assign variable length bit codes to the data that is being compressed. The characters are assigned codes based on their frequency of occurrence and the characters with highest frequency are assigned smallest length bit codes and the characters with least frequency are assigned the longest length bit codes.

Every 24 bit .bmp file starts with a 54 byte header, the 18th byte specifies the width of the image and the 22nd byte specifies the height of the image. After the header the color of each pixel is specified by its RGB values. The color of the bottom right pixel is stored first. The colors are stored in the order of blue, green and red and each color takes 1 byte each. Therefore, each pixel requires 3 bytes.

Algorithm

1. Read .bmp image and store its width, height, size and information of pixels in imageInfo struct.
2. The information of each pixel is saved in an object of Pixel class containing information about the RGB values of the pixel, and the corresponding hex values.
3. No. of distinct colors are calculated and the frequency of each distinct color and its hex value is stored in an object of class color.
4. A Huffman tree is created using the information in step 3 and the bit codes are saved with their respective hex values and frequency in the color class.
5. The header of the new file is written
6. The entire Image Matrix is traversed and the corresponding bit code of the color of each pixel is written in the file.
7. The written file is saved with an extension of .cmpbmp.

Proposed File Format (.cmpbmp)

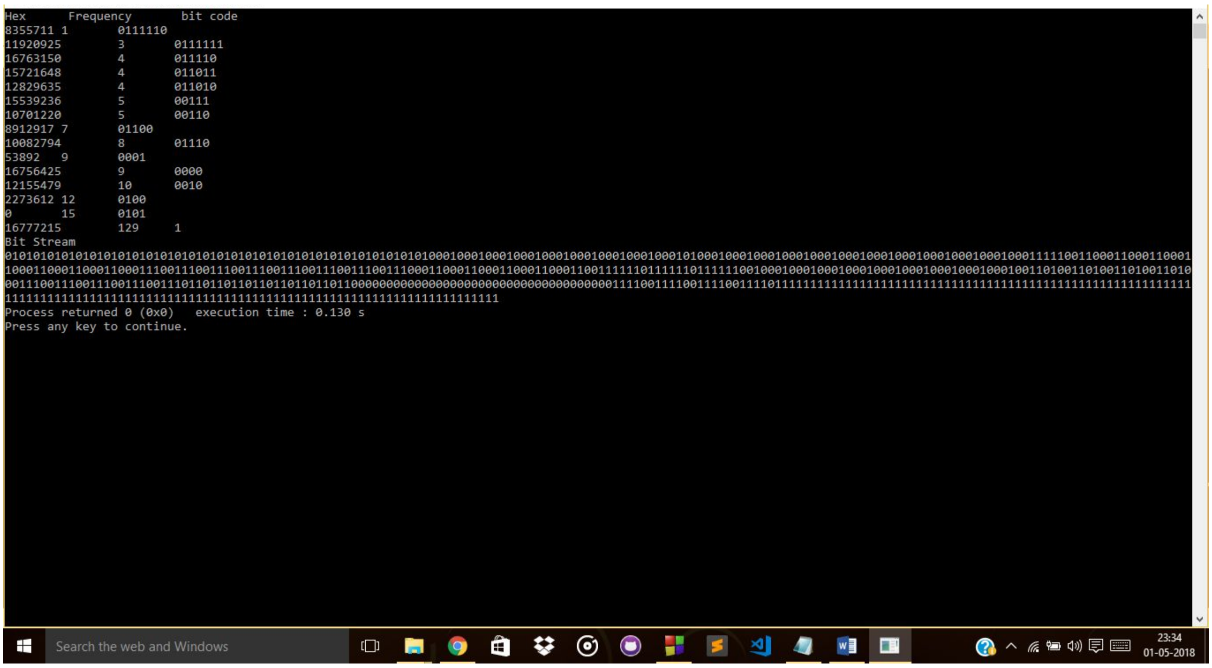
The first 4 bytes of the file contain the number of distinct colors in the compressed image. The information of hex value and its corresponding frequency is also stored for each distinct color where both hex value and frequency takes 4 bytes each.

The actual image contains the bit codes generated by Huffman coding and is the actual compressed data.

Using the information in header a Huffman tree is created and bit codes are generated which are used to decompress the image.

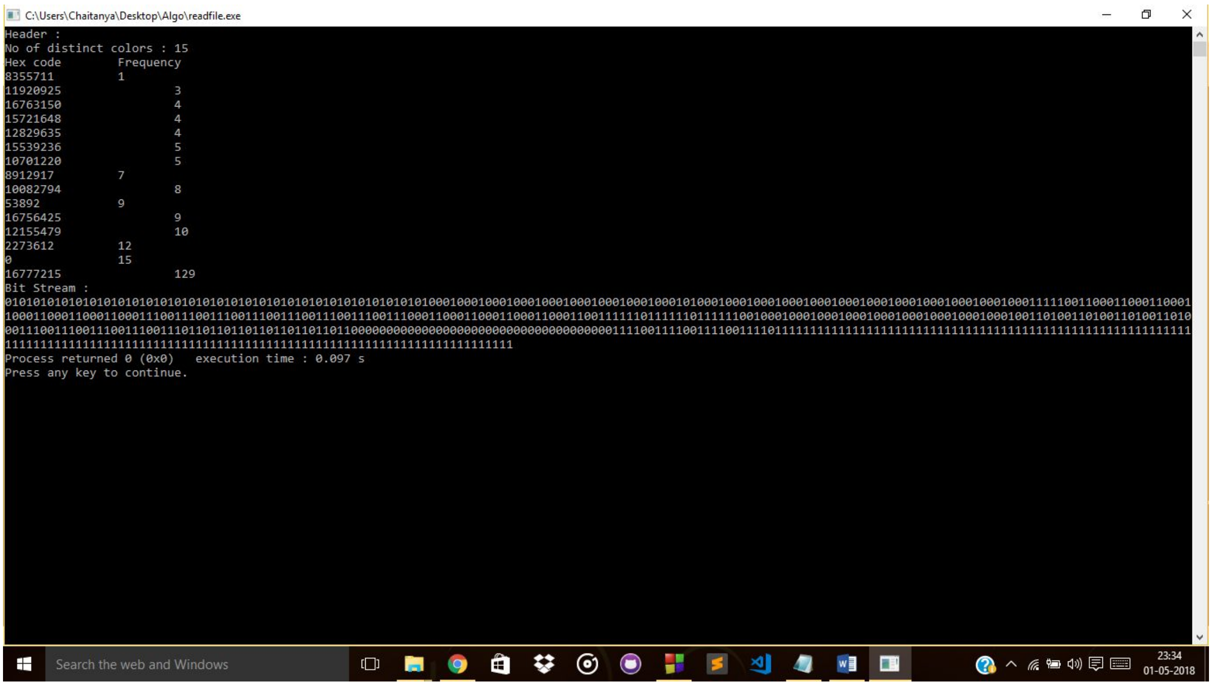
Screenshots

Input



```
Hex      Frequency       bit code
8355711 1       0111110
11920925        3       0111111
16763150        4       011110
15721648        4       011011
12829635        4       011010
15539236        5       00111
10701220        5       00110
8912917 7       01100
10082794        8       01110
53892   9       0001
16756425        9       0000
12155479        10      0010
2273612 12      0100
0       15      0101
16777215        129     1
Bit Stream
010101010101010101010101010101010101010101010101010101010101010100010001000100010001000100010001000101000100010001000100010001000100010001000100011111001100011000110001
100011000110001100011100111001110011100111001110011100111001110011000110001100011000110011111101111110111111001000100010001000100010001000100010001000110100110100110100110100110
001110011100111001110011101101101101101101101100000000000000000000000000000000000000001111001111001111001111011111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111
Process returned 0 (0x0)   execution time : 0.130 s
Press any key to continue.
```

Output


C:\Users\Chaitanya\Desktop\Algo\readfile.exe
Search the web and Windows
23:34
01-05-2018

```
Header :
No of distinct colors : 15
Hex code        Frequency
8355711         1
11920925                3
16763150                4
15721648                4
12829635                4
15539236                5
10701220                5
8912917         7
10082794                8
53892           9
16756425                9
12155479                10
2273612         12
0               15
16777215                129
Bit Stream :
010101010101010101010101010101010101010101010101010101010101000100010001000100010001000100010001010001000100010001000100010001000100010001000100011111001100011000110001
100011000110001100011100111001110011100111001110011100111001110001100011000110001100011000110011111101111110111111001000100010001000100010001000100100110100110100110100110
001110011100111001110011101101101101101101101101100000000000000000000000000000000000001111001111001111001111011111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111
Process returned 0 (0x0)   execution time : 0.097 s
Press any key to continue.
```

References

1. http://www.di.unito.it/~marcog/SM/BMPformat-Wiki.pdf
2. https://www.geeksforgeeks.org/greedy-algorithms-set-3-huffman-coding/
3. https://www.geeksforgeeks.org/unordered_map-in-stl-and-its-applications/

Division of Work

Utkrisht – File reading and writing and debugging.

Chaitanya – Frequency calculation and distinct colors and debugging.

Aarya – Huffman Coding.