Huffman Project Report

Aya Abdullah
ayaabdulla
ayaabdullahfarag1999@gmail.com

Khloud Abdelazeem
Khloud-azeem
Khloud.mostfa00@eng-st.cu.edu.eg

Filobater george
FilobaterGeorge
filobater.khedewy@eng-st.cu.edu.eg

Meirna Kamal
Meirna-kamal
meirna.abdelrahman99@eng-st.cu.edu.eg

Yasser Nasser
yasser1412
yassernasser1412@gmail.com
yasser.abdallah98@eng-st.cu.edu.eg

***Abstract*—This document is a report introducing our huffman project including all the resources we used, a block diagram describing our systems, all our steps to reach our running program, A user manual for the system, and also screenshots for our running program.**

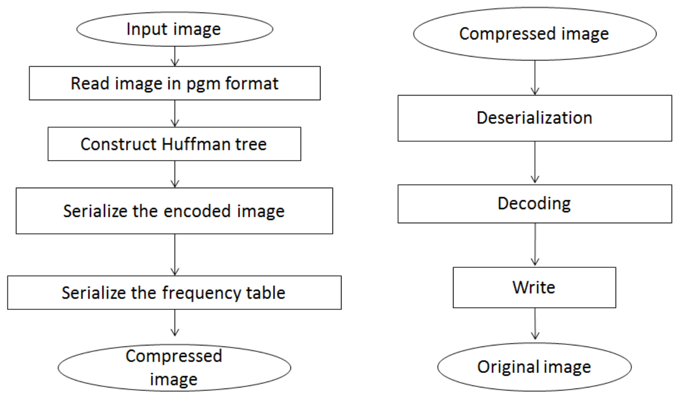
I. Introduction

Although hard drives keep getting bigger and bigger as time passes, The files we want to store are also getting bigger within time, which makes even today's hard disks seem small, so this makes us always look forward to new and better ways to store data with the least space as possible. One of the best techniques to save more storage and to use our storage better is to compress files. There are many known compression algorithms like (JPEG or MP3) , but most of these algorithms are lossy, which means that by decompressing back the file they don't create a perfect copy of the original file because their compressing mechanism depends on summarizing the data. Huffman compression, on the other hand, which is a great technique to compress and shrink the sizes of texts, images, and other files, is a " lossless data compression" which means that it doesn't lose any of the data while coding, which makes it of great importance. It is mainly based on the frequency of occurrence of a character or a symbol in the file being compressed. We will be using Huffman algorithm in this project to compress and decompress an image.

II. Motivation

The reason behind us choosing this particular project was the interesting and useful techniques that we would learn while coding it. It was our first time dealing with images in a code, starting from reading the image until compressing this image. What interested us the most was the process of converting the pgm file to an array of values of the intensity of each pixel, and how we would encod this array using a huffman tree to a binary file and reconstruct it that the storage of each pixel isn't fixed, but variable according to the frequency count of the values, that the most frequent pixels values occupy the least number of bits. We also benefited a lot from creating a graphical user interface (GUI) to integrate our functionalities by using QT creator.

III. Block diagram



The previous figure shows a block diagram describing the compression and decompression proccess of our system

IV. Resources

Libraries used are: iostream, string, fstream, sstream, bits/stdc++.h,QFile,QFileDialog,QTextStream,QMessageBox

Tools used are: Vs code, Qt creator

V. Challenges and problems

We found some difficulties during coding that we were able to solve after a lot of searching,as The GUI was a challenging part. The link between the code and interface made using qt required several widgets and codes that we had to search for and learn. We had problems in the files paths and browsing them from the machine and to deal with.
Storing each pixel with its frequency and particular code was also another difficulty, Converting the data of the file to binary form and storing them was also challenging.We also struggled to go through the huffman tree while decompression.

VI. Contributions

All 5 of us worked on the idea as a team and tried to solve the problems that faced us together but mainly each one of us focused on one thing as the following

Aya abdullah– Read and write of the code

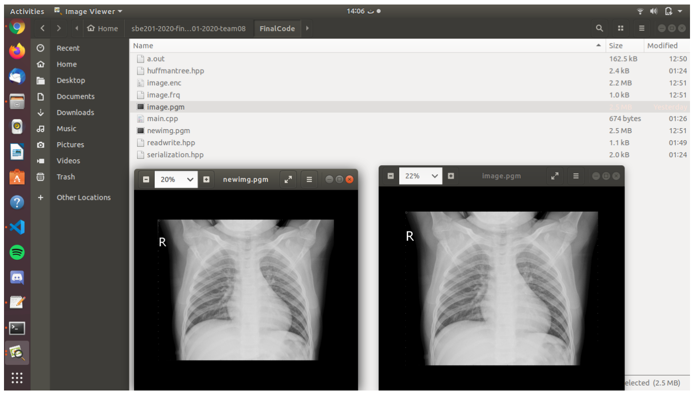
Filobater george and yasser nasser – Encoding
Yasser Nasser – Decoding

Khloud abdelazeem – Serialization and Deserialization

Meirna kamal and Filobater George– GUI
Meirna Kamal – Latex Document

VII. Results

These figures are the outputs of our running program

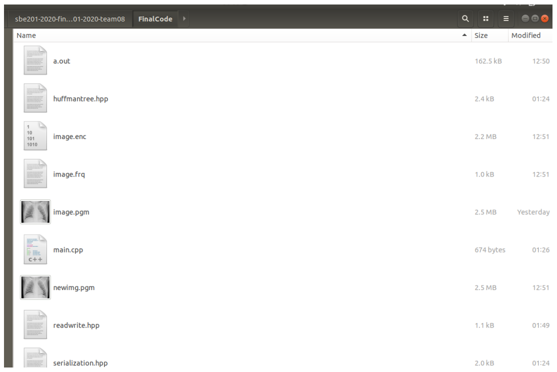


Activities · Image Viewer · 14:06

Home › sbe201-2020-fin...01-2020-team08 › FinalCode

Name	Size	Modified
a.out	162.5 kB	12:50
huffmantree.hpp	2.4 kB	01:24
image.enc	2.2 MB	12:51
image.frq	1.0 kB	12:51
image.pgm		
main.cpp	674 bytes	01:26
newimg.pgm	2.5 MB	12:51
readwrite.hpp	1.1 kB	01:49
serialization.hpp	2.0 kB	01:24

The previous screenshots shows the difference between the original photo (img.pmg) and the compressed photo (newimg.pmg)



sbe201-2020-fin...01-2020-team08 › FinalCode

Name	Size	Modified
a.out	162.5 kB	12:50
huffmantree.hpp	2.4 kB	01:24
image.enc	2.2 MB	12:51
image.frq	1.0 kB	12:51
image.pgm	2.5 MB	Yesterday
main.cpp	674 bytes	01:26
newimg.pgm	2.5 MB	12:51
readwrite.hpp	1.1 kB	01:49
serialization.hpp	2.0 kB	01:24

The previous screenshot shows us the two enc and frq files that are the product of our compression, and that are also used to decompress and get our original image.

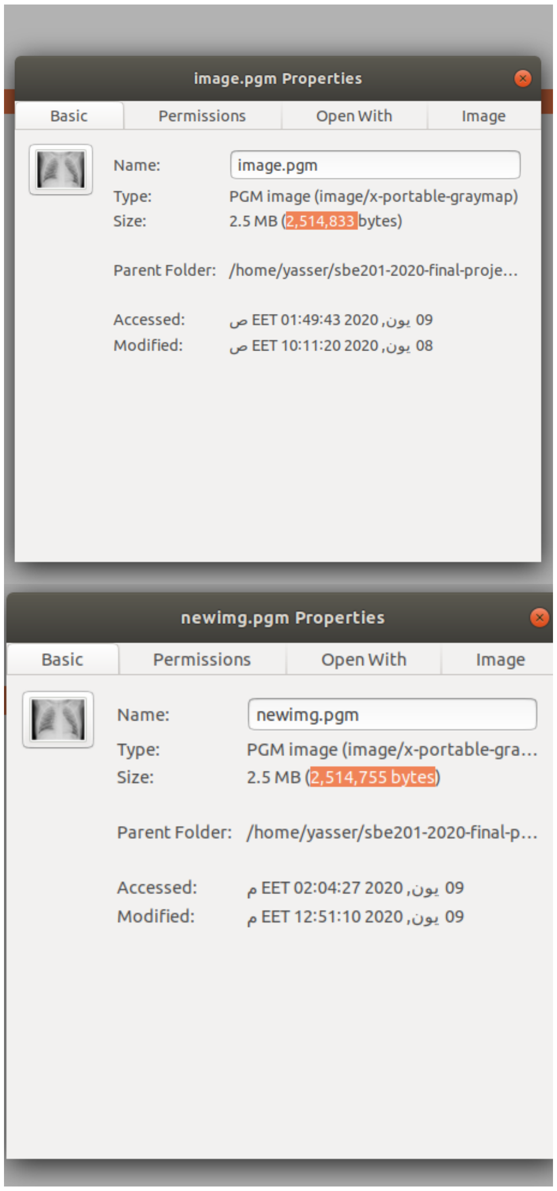


image.pgm Properties

Basic | Permissions | Open With | Image

Name: image.pgm
Type: PGM image (image/x-portable-graymap)
Size: 2.5 MB (2,514,833 bytes)

Parent Folder: /home/yasser/sbe201-2020-final-proje...

Accessed: 09 يون, 2020 01:49:43 EET ص
Modified: 08 يون, 2020 10:11:20 EET ص

newimg.pgm Properties

Basic | Permissions | Open With | Image

Name: newimg.pgm
Type: PGM image (image/x-portable-gra...
Size: 2.5 MB (2,514,755 bytes)

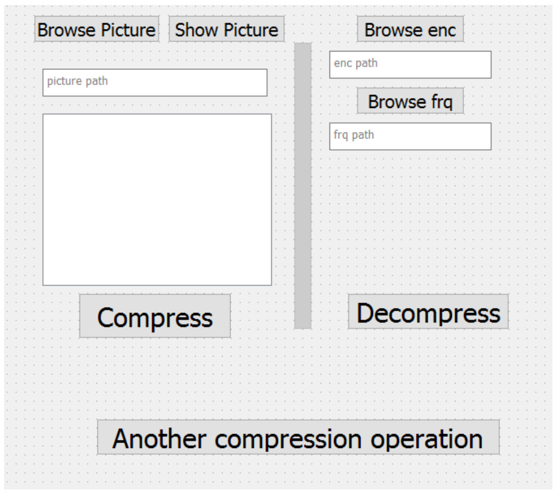
Parent Folder: /home/yasser/sbe201-2020-final-p...

Accessed: 09 يون, 2020 02:04:27 EET م
Modified: 09 يون, 2020 12:51:10 EET م

The previous two pictures show us that the image's size has become smaller and the image is compressed

VIII. User manual for the system

As shown in the following figure the user will be able to simply upload a photo by clicking on (browse picture) as it will take the path of the picture the user chooses, then click on show photo so the chosen photo will appear in the blank box. After clicking on the compress button the user will be able to choose the location where he wants to save the frequency table and the encrypted file.
In order to decompress the image the user will simply click on browse enc and browse frq to get their location, and click on decompress button which will allow the user to choose the location that he wants to save the decompressed image.



References

[1] https://www.geeksforgeeks.org/huffman-coding-greedy-algo-3/
[2] https://www.programiz.com/dsa/huffman-coding
[3] https://www.youtube.com/watch?v=eP3FzXc1K3g
[4] https://www.youtube.com/watch?v=LplgowaCY4c
[5] https://www.youtube.com/watch?v=Y5SWDAIZ0sQ
[6] https://www.youtube.com/watch?v=GxlB34Cn0zwt=1s
[7] 1. Paulus, D. (2012). Pattern Recognition and Image Processing in C++. Vieweg+ Teubner Verlag.
[8] Frery, A. C., Perciano, T. (2013). Introduction to image processing using R: learning by examples. Springer Science Business Media
[9] Paulus, D., Hornegger, J. (2003). Applied pattern recognition: algorithms and implementation in C++. Springer Science Business Media.
[10] https://www.geeksforgeeks.org/image-compression-using-huffman-coding/: :text=Huffman
[11] https://nirav.com.np/2019/02/14/writing-huffman-compression-in-cpp.html
[12] https://github.com/practicingruby/binary$_examples http : //www.cplusplus.com/reference/string/string/erase/$
[13] https://stackoverflow.com/questions/6038718/convert-integer-to-bits
[14] https://stackoverflow.com/questions/5590381/easiest-way-to-convert-int-to-string-in-c
[15] https://doc.qt.io/qt-5/qtwidgets-tutorials-addressbook-part6-example.html
[16] https://www.geeksforgeeks.org/stdstringreplace-stdstringreplace$_if - c/https : //stackoverflow.com/questions/12992606/qt - load - an - image - from - the - computer - in - gui/12992686$
[17] https://www.qtcentre.org/threads/14579-How-to-save-file-with-QFileDialog