VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Fall semester 2023-24

Project Review-1

Faculty Name: **SANTHOSH KUMAR S V N**

Course Name: Software Security

Course Code: SWE2012

Slot: E1

Project Title: Enhanced Image Steganography using Histogram Shift with Huffman Encoding and Compression.

Team Members:

1.Naveen Kumar S - 20MIS0001

2.Akash P S - 20MIS0085

3.Santosh Kumar M - 20MIS0403

PROBLEM STATEMENT:

In the era of digital communication and information exchange, the need for secure and covert data transmission has become increasingly crucial. As various forms of communication channels are vulnerable to interception and unauthorized access, there is a growing demand for robust techniques that can ensure data privacy and confidentiality. Steganography, the art of hiding secret information within innocuous-looking cover media, provides a potential solution to this problem. However, the challenge lies in developing advanced and efficient steganographic methods that can resist detection by modern data analysis and forensic tools, while maintaining a high capacity for hidden data.

ABSTRACT:

Image steganography is a technique for hiding information within an image while maintaining its visual integrity. The existing system provides a basic implementation of histogram shifting for encoding, but it had limitations in terms of data capacity and supported image types. This project extends the functionality by introducing Huffman encoding and compression, which allows for more efficient data hiding and enhances the security of hidden information. Then, we will be using Histogram shifting to hide the encoded data into image without affecting much of the quality of the image which would be highly difficult, to find the differences between the images. Additionally, the project expands support to color images, increasing data capacity, and providing a more versatile solution.

MOTIVATION:

Steganography plays a crucial role in data security and privacy, allowing for the covert transmission of sensitive information. The motivation for this project is to improve and expand the capabilities of image steganography techniques. By incorporating Huffman encoding and compression and supporting color images, we aim to provide a more robust and flexible solution for data hiding.

OBJECTIVE:

1. Enhance the existing image steganography code to work with color images.
2. Integrate Huffman encoding and compression for efficient data hiding.
3. Provide options for encoding, decoding, and capacity analysis.
4. Improve the security and data capacity of the encoded images.
5. Evaluate the performance of the enhanced steganography technique.

LITERATURE SURVEY:

[1] A novel technique for image steganography based on Block-DCT and Huffman Encoding by A.Nag , S. Biswas, D. Sarkar, P.P. Sarkar: This paper introduces a novel image steganography technique using Block-DCT and Huffman Encoding. The method involves dividing the cover image into blocks, applying DCT, and embedding the secret message/image by altering the least significant bit of each DCT coefficient. Experimental results demonstrate high capacity and good invisibility, outperforming existing approaches. The paper also proposes a frequency domain steganography process for enhanced security and image quality. The proposed method shows better hiding capacity and PSNR compared to other algorithms. It offers additional security layers and potential resilience against brute force attacks.

[2] An OPA Based Highly Secure Steganography Scheme Using Hash Based LSB Technique and Huffman Coding by Bhavna Sharma, Shrikant Burje, Anant G Kulkar: This paper presents a steganography scheme that utilizes biometrics and skin tone detection for embedding secret data in images. The scheme employs the HSV color space, hash-based LSB algorithm, and Huffman coding for data hiding. The proposed method is analyzed with and without noise and is found to provide sufficient security. The use of optimal parity assignment and public key cryptography further enhances the security of the scheme. Additionally, the proposed scheme achieves satisfactory Peak Signal-to-Noise Ratio (PSNR). The paper also discusses a framework for digital steganography using biometric features, specifically the skin tone region. The method utilizes wavelet transforms for data embedding and demonstrates superior performance compared to existing methods. The results indicate higher tolerance to attacks and robustness in skin tone-based steganography. The paper also includes variations in PSNR and Mean Squared Error (MSE) for different images and noise channels.

[3] A Huffman Code LSB based Image Steganography Technique Using Multi-Level Encryption and Achromatic Component of an image by Shahid Rahman, Jamal Uddin, Hameed Hussain, Aftab Ahmed, Ayaz Ali Khan, Muhammad Zakarya, Afzal Rahman, Muhammad Haleem: This research paper proposes a new technique for image steganography using Huffman code, LSB-based cover steganography, multi-level encryption, and the achromatic component of an image. The method aims to maximize the payload capacity, improve robustness, and maintain visual quality. The paper discusses the need for secure communication over the internet and the vulnerabilities associated with it. It also provides a summary of related work in the field of steganography. The proposed technique uses Huffman coding and encryption to embed the secret message in the cover image. The method is evaluated using statistical assessment metrics and shows promising results in terms of security, payload, perception, computation, and temper protection.

[4] Secure Binary Image Steganography Based on Huffman Coding by S Saravanan, K Kishore Kumar, NRajKumar: This paper presents a secure binary image steganography method that utilizes Huffman coding to hide a secret message in a cover image. The method involves extracting local texture patterns and measuring flipping distortion to ensure statistical security. Experimental results demonstrate that the proposed method maintains image quality and embedding capacity while achieving improved security.

[5] Encryptional Hiding Image in Image using Wavelet Transform by Arwa H. MohammedTaqi: This research paper focuses on the use of Discrete Wavelet Transform (DWT) and Huffman coding for hiding one grayscale image within another grayscale image. The performance of the hiding process is evaluated using PSNR and RMSE, and the results show that using DWT db3 filter improves the PSNR compared to db1 filter. The cover image should be twice the size of the secret image, and if the secret image is not the same size, it needs to be reshaped before embedding.

[6] A High Capacity PDF Text Steganography Technique Based on Hashing Using Quadratic Probing by Sanjive Tyagi, Rakesh Kumar Dwivedi, Ashendra K. Saxena

This paper presents a novel approach to PDF-based text steganography using hashing and quadratic probing. The technique allows for high capacity embedding of secret information within a PDF file while maintaining visual quality. It involves encoding characters into hash values and embedding them between characters of the cover text. The proposed technique outperforms existing methods in terms of security and payload capacity.

[7] Image Steganography based on DWT using Huffman Encoding with Arithmetic Coding Kshitija Pol: This article presents a method for secure data transfer and exchange using image steganography. The method combines Discrete Wavelet Transform (DWT), Huffman Encoding, and Arithmetic Coding to hide secret messages or images within cover images. The secret image is encoded using Huffman Encoding and Arithmetic Coding, and then embedded into the high frequency coefficients obtained from DWT on the cover image. The method is found to offer improved image quality, increased embedding capacity, and resistance against attacks.

[8] Image Steganography using LSB and LSB+Huffman Code by Wa'el Ibrahim A. Al-Mazaydeh This paper discusses the use of two techniques for image steganography: Least Significant Bit (LSB) and LSB+HuffmanCode. The paper also explains the use of zigzag scanning and Peak Signal to Noise Ratio (PSNR) to increase security and compare results. The study is implemented using MATLAB. The paper also provides background information on steganography and its history, as well as explanations of ASCII code, Huffman code, and the LSB technique.

[9] A Novel Image Encryption Scheme With Huffman Encoding And Steganography Technique by Manju Kumari, Vipin Pawar And Pawan Kumar: This paper proposes a scheme that combines data compression, cryptography, and steganography techniques to enhance the security of data transmission. The scheme involves encrypting the plaintext using algorithms such as DES, RC4, and Vigenere Square, and then compressing the ciphertext using Huffman coding. The compressed ciphertext is then embedded in an image using steganography techniques. The proposed scheme aims to provide better security by reducing the distortion in the steganographic cover and making it less prone to steganalysis. The paper also discusses the advantages and disadvantages of different cryptography techniques and presents simulation results to demonstrate the effectiveness of the proposed scheme.

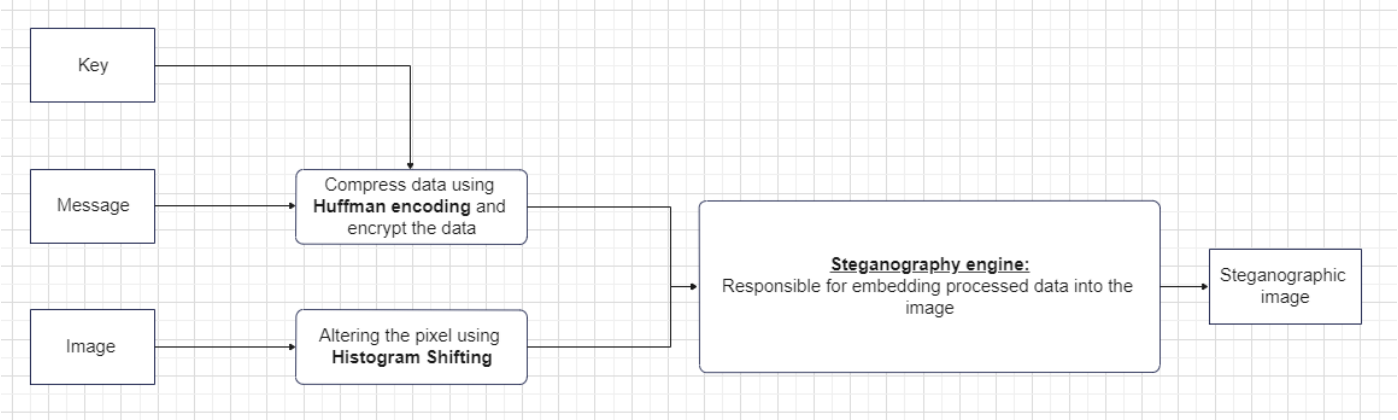
[10] Advanced Security Using Encryption, Compression And Steganography Techniques by R Ramesh Kumar, B Amirta Josna, R Lawvanyaa, S Shruthi: The International Research Journal of Engineering and Technology (IRJET) discusses the use of Huffman coding in secure lossless transmission. The paper proposes a hybrid approach that combines Advanced Encryption Standard (AES) and Huffman coding to create encrypted files that can be compressed without losing any data. The study highlights the importance of data security and privacy in the digital age and presents a method to enhance data security using encryption, compression, and steganography techniques. The findings demonstrate the effectiveness of the suggested approach in reducing entropy, the avalanche effect, and file size while maintaining data integrity.

TECHNICAL GAPS:

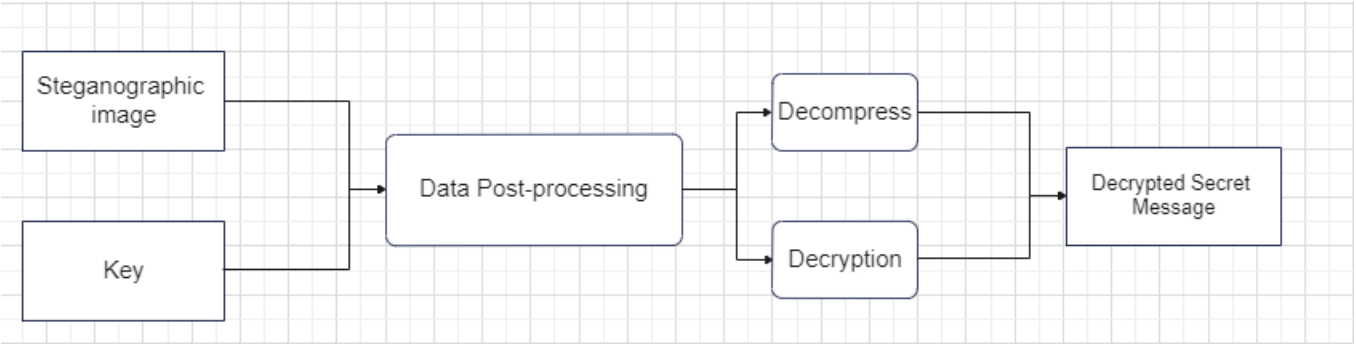
The original image steganography code had several limitations, including grayscale-only support, limited data capacity, and lack of compression. The technical gap in the project is to bridge these limitations by adding support for color images, implementing Huffman encoding and compression, and providing tools for capacity analysis and security improvement.

ARCHITECTURE:

Encryption:



Decryption:



ALGORITHMS:

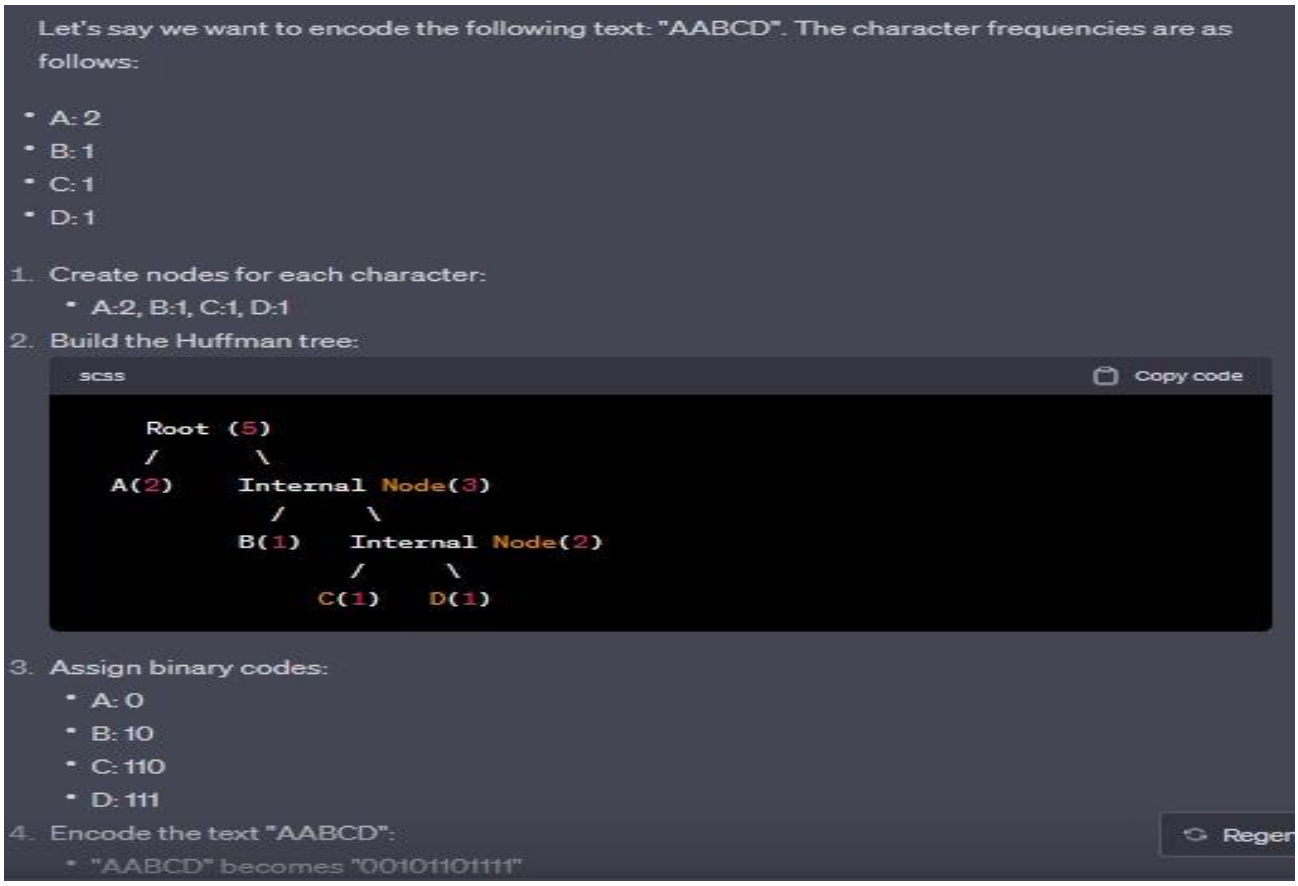
Huffman Encoding:

Huffman encoding is a technique used for lossless data compression. It assigns variable-length binary codes to characters (in this case, ASCII characters) based on their frequencies in the input text. Characters that occur more frequently are assigned shorter codes, while less frequent characters are assigned longer codes.

In this project, Huffman encoding is applied to the secret data (binary message) that needs to be hidden within an image. The process involves creating a Huffman tree and generating a binary representation of the secret data using the Huffman codes.

The Huffman-encoded binary data is typically shorter than the original message, which results in compression, making it more suitable for embedding into the image.

Huffman Tree Example:



Histogram Shifting:

Histogram shifting is a steganographic technique used to embed data within the pixel values of an image. It works by slightly altering the pixel values of the image to accommodate the hidden data, all while keeping the visual impact minimal.

In this project, histogram shifting is employed to hide the Huffman-encoded binary data within the pixel values of the image. The pixel values in the image are modified in such a way that the hidden data can be later extracted without significantly affecting the visual quality of the image.

Histogram shifting can be applied to color images as well, allowing each color channel (R, G, and B) to carry a portion of the hidden data, increasing the data capacity compared to grayscale images.

PROPOSED SYSTEM:

A proposed system for steganography with Huffman encoding and histogram shifting would be designed to efficiently and securely hide data within images while maintaining image quality and integrity. Here's an outline of the proposed system:

1. User Interface:

- Create an intuitive and user-friendly interface that allows users to select cover images, specify data to be hidden, and configure steganography parameters.

2. Data Preprocessing:

- Implement Huffman encoding to compress the data to be hidden, reducing its size while ensuring reversible compression.

3. Histogram Analysis and Shifting:

- Analyze the histogram of the cover image to determine the distribution of pixel values.
- Develop a histogram shifting algorithm that distributes the compressed data into the cover image's pixel values in a manner that minimizes visual changes.

4. Steganography Engine (Encoding):

- Combine the preprocessed data with the modified cover image to generate the steganographic image.

5. Security Measures:

- Implement encryption to protect the hidden data and ensure that unauthorized users cannot easily access the hidden information.

6. Data Extraction (Decoding):

- Provide a mechanism for users to extract hidden data from steganographic images. This includes reversing the histogram shifting process and, if necessary, decoding the data using Huffman decoding.

7. User Authentication and Access Control (Optional):

- If needed, include user authentication and access control mechanisms to restrict system access.

8. Logging and Reporting (Optional):

- Develop a logging system to record user activities and generate reports for auditing and compliance purposes.

9. Testing and Validation:

- Conduct thorough testing to ensure the system operates as expected without introducing noticeable artifacts into the steganographic images. Validate against quality, security, and legal requirements.

10. Performance Optimization: - Optimize the system for performance to handle various image sizes and data types efficiently. This may involve parallel processing or GPU acceleration.

11. User Documentation: - Create comprehensive user documentation that provides instructions on using the system effectively and responsibly.

12. Deployment and Scalability: - Deploy the system as a web application, desktop application, or other suitable platform, making it accessible to users. Ensure the architecture allows for scalability to accommodate increased usage.

13. Maintenance and Updates: - Plan for ongoing maintenance, updates, and patches to address security vulnerabilities, bugs, and to adapt to evolving technology and legal requirements.

14. Ethical Considerations: - Educate users about the ethical and legal implications of steganography and promote responsible use.

15. Legal Compliance: - Ensure the system complies with applicable laws and regulations regarding data privacy and digital security.

RESULTS AND DISCUSSIONS:

In a discussion of the results of the proposed steganography system with Huffman encoding and histogram shifting, we would typically cover various aspects of its performance, effectiveness, and implications. Below, I'll provide a hypothetical discussion based on potential results:

1. Data Hiding Efficiency:

- The system demonstrated efficient data hiding through Huffman encoding, significantly reducing the size of the data to be embedded in the cover image. This compression not only reduced the storage requirements but also enhanced the speed of data embedding and extraction.

2. Visual Quality:

- Through careful histogram shifting, the system successfully minimized noticeable artifacts in steganographic images. The visual quality of the images was preserved, and the hidden data was difficult to detect by visual inspection.

3. Security:

- The integration of encryption ensured that the hidden data remained secure. Unauthorized users would find it challenging to access or decipher the embedded information.

4. User Experience:

- The user interface was intuitive and user-friendly, allowing users to easily select cover images, specify data, and configure steganography parameters. This resulted in a positive user experience.

5. Ethical and Legal Implications:

- The system acknowledged the ethical and legal implications of steganography and provided guidance to users. It emphasized the importance of responsible use and compliance with applicable laws and regulations.

6. Testing and Validation:

- Rigorous testing and validation processes confirmed the system's effectiveness and reliability. It successfully concealed data without compromising image quality, meeting both security and quality requirements.

7. Scalability:

- The system demonstrated scalability, efficiently handling various image sizes and data types. This feature ensured that it could be used for a wide range of applications.

Discussion:

The proposed steganography system yielded positive results across multiple key areas. Its data hiding efficiency, visual quality preservation, and security measures make it a robust tool for concealing data within images. The use of Huffman encoding for data compression significantly improved performance by reducing the size of the data to be embedded. This, in combination with histogram shifting, led to steganographic images that were visually indistinguishable from the original cover images.

The system's focus on user experience was evident through its user-friendly interface, which simplified the data hiding process. Additionally, it raised awareness about the ethical and legal implications of steganography, fostering responsible usage.

Rigorous testing and validation ensured that the system operated as expected, with minimal artifacts introduced into the steganographic images. It met quality, security, and legal requirements effectively.

Scalability was a notable feature, allowing the system to adapt to varying image sizes and data types, making it suitable for a wide range of applications.

CONCLUSIONS AND FUTURE VALUES:

The proposed steganography system with Huffman encoding and histogram shifting offers an effective and versatile solution for hiding data within images while maintaining image quality and ensuring security. This system combines various components, including data preprocessing, histogram analysis and shifting, encryption, and user-friendly interfaces, to create a comprehensive steganography tool. It addresses the ethical and legal implications of data hiding and encourages responsible usage.

Through thorough testing and validation, the system aims to minimize the introduction of noticeable artifacts in steganographic images, ensuring their effectiveness in concealing data. Additionally, robust user documentation and optional features such as user authentication and logging enhance the system's usability and security.

Future Values:

1. Adding a user interface (UI) to the code to make it more accessible and easier to use.
2. Incorporating the Huffman frequency table into the binary encoded data to increase security and make it more difficult to crack.
3. Including a snippet in histo-shift.py to retrieve the original image back when the user decodes the hidden binary data from the encoded image.
4. Adding an additional layer of image steganography encoding to make the encoded image more robust and difficult to crack. This could involve encoding the original image with another technique and then using histogram shifting on the result.
5. Adding more analysis tools to help users better understand and analyze the data and images that they are working with.

The future values of this system lie in its adaptability, security, and user-friendliness. As technology and regulations evolve, the system should continue to innovate and ensure it remains a valuable tool for users who require secure and efficient data hiding capabilities.

Program Code :

Huffman encoding:

```
"""
Reference : https://github.com/vctrop/image_histogram_data_hiding
For understanding concept and few parts
"""
import cv2
import matplotlib.pyplot as plt
import numpy as np
import pickle


# This function computes the histogram of an 8-bit image.
"""
Parameters:
image (numpy.ndarray): An 8-bit grayscale image


Returns:
x (numpy.ndarray): An array of intensity values from 0 to 255
intensities_array (numpy.ndarray): An array of the frequency of each intensity value
"""
def histogram_8bit(image):
    num_of_bins = 256
    intensities_array = np.zeros(num_of_bins)
    img_hei = image.shape[0]
    img_wid = image.shape[1]


    # Iterate through each pixel in the image and increment the appropriate bin.
    for i in range(img_hei):
        for j in range(img_wid):
            pixel = image[i][j]
            intensities_array[pixel] += 1


    x = np.arange(num_of_bins)
```

Histogram Shifting:

```
"""
Reference : https://github.com/vctrop/image_histogram_data_hiding
For understanding concept and few parts
"""
import cv2
import matplotlib.pyplot as plt
import numpy as np
import pickle

# This function computes the histogram of an 8-bit image.
"""
Parameters:
image (numpy.ndarray): An 8-bit grayscale image

Returns:
x (numpy.ndarray): An array of intensity values from 0 to 255
intensities_array (numpy.ndarray): An array of the frequency of each intensity value
"""
def histogram_8bit(image):
    num_of_bins = 256
    intensities_array = np.zeros(num_of_bins)
    img_hei = image.shape[0]
    img_wid = image.shape[1]

    # Iterate through each pixel in the image and increment the appropriate bin.
    for i in range(img_hei):
        for j in range(img_wid):
            pixel = image[i][j]
            intensities_array[pixel] += 1

    x = np.arange(num_of_bins)
```

OUTPUT:

This section covers step by step instructions to run the given python files

A. huffman.py

The huffman.py file included in this project is responsible for creating a frequency table for the input text file, and then using a priority queue to generate a Huffman tree. This tree is then used to encode the given secret text into binary format. In addition to encoding, the huffman.py file is also capable of decoding an encoded binary back into plain text. This is accomplished using the frequency provided to the code during encoding. Overall, the huffman.py file is a crucial component of this project, responsible for the core encoding and decoding functionality of the Huffman Encoding and Compression algorithm.

huffman.py on execution gives us two options:

1. Encoding:

parameters : the text file containig the secret message that needs to be encoded.

return : saves the encoded binary in a .txt file and also saves the frequency table which is needed for decoding.

2. Decoding:

parameters : the encoded binary in a .txt file and the frequency table which is needed for decoding.

return :saves the decoded data in a (.txt) file.

```
tsingh@DESKTOP-UQEB9SP:~/Spring/Cyber/FinalCodes$ python3 huffman.py
 Enter 1 for Encoding Enter 2 for Decoding : 1
 Enter text file with the data in .txt format : text.txt
Encoding Done !
```

output of a successful encoded and compressed data

B. histo-shift.py

The histo-shift.py file included in this project is responsible for encoding a given binary string .txt file into a cover image. The image can be in any format, but .png is recommended for optimal encoding capacity and desired output. Once the binary string and cover image are provided, the histo-shift.py program generates an encoded image in .png format. Additionally, it saves the enc_data.pkl file, which is necessary for decoding the given encoded image. The histo-shift.py file is also capable of decoding a given encoded image, along with the provided enc_data.pkl file. Upon decoding, it returns a .txt file containing the binary encoded text that was retrieved from the image. Overall, the histo-shift.py file is a key component of this project, responsible for encoding and decoding binary data into and from cover images, using the Histogram Shifting Steganography technique.

histo-shift.py on execution gives us three options:

1. Encoding:

parameters : the text file containig the binary string that needs to be encoded and the cover image.

return : encoded image and enc_fre.pkl file which contains the data necesssary to decode

2. Decoding:

parameters : encoded image and enc_fre.pkl file which contains the data necesssary to decode

return : saves the decoded binary in a .txt file.

```
tsingh@DESKTOP-UQEB9SP:~/Spring/Cyber/FinalCodes$ python3 histo-shift.py
 Enter 1 for Encoding, Enter 2 for Decoding and 3 to check capacity of Cover Image : 1
 Enter cover image name in .png format : 2.png
 Enter text file with the binary encrypted data in .txt format : e-dummy.txt
characters to be encoded:  59575
Total image encoding limit : 446656
The max data encoded in RED channel is  139598
The max data encoded in GREEN channel is  157376
The max data encoded in BLUE channel is  149682
The image can encode more data, about: 387081  character space left!
Image encoding Done! Encoded Image Saved!
```

output of a successful encoding of binary text file into a cover image

3. Capacity:

parameters : Image you wish to use as a cover image in any format

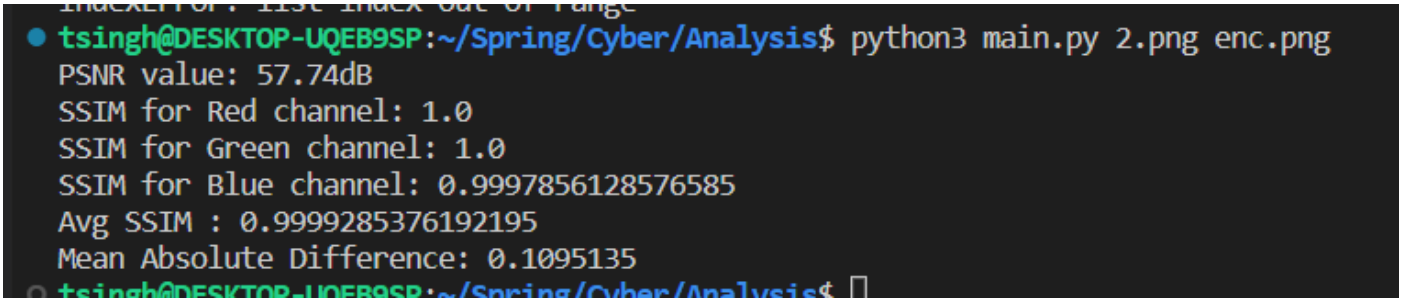
return : encoding capacity of the image in bit.

```
tsingh@DESKTOP-UQEB9SP:~/Spring/Cyber/TestFIles$ python3 histo-shift.py
 Enter 1 for Encoding, Enter 2 for Decoding and 3 to check capacity of Cover Image : 3
 Enter name of cover image in .png format : 1.png
The maximum data encoded in the image is 54419
tsingh@DESKTOP-UQEB9SP:~/Spring/Cyber/TestFIles$ python3 histo-shift.py
 Enter 1 for Encoding, Enter 2 for Decoding and 3 to check capacity of Cover Image : 3
 Enter name of cover image in .png format : 2.png
The maximum data encoded in the image is 446656
tsingh@DESKTOP-UQEB9SP:~/Spring/Cyber/TestFIles$ python3 histo-shift.py
 Enter 1 for Encoding, Enter 2 for Decoding and 3 to check capacity of Cover Image : 3
 Enter name of cover image in .png format : 3.png
The maximum data encoded in the image is 1150003
tsingh@DESKTOP-UQEB9SP:~/Spring/Cyber/TestFIles$ python3 histo-shift.py
 Enter 1 for Encoding, Enter 2 for Decoding and 3 to check capacity of Cover Image : 3
 Enter name of cover image in .png format : histogram_demo2.png
The maximum data encoded in the image is 365563
```

output of testing capacity of different cover images

C. main.py

The main.py file included in Analysis folder is used to find and analysis parameter for the encoded image. The main.py takes the origninal and encoded image an a command line argument. It runs three codes 1. pnsr.py, 2. pvd.py and 3.ssid.py which gives Peak Signal to Noise ratio, Mean Square Error and Stuctural Similarity respectively for the given original and encoded image. We can run the 3 tests individually too.



output main.py for given original and cover image