IMAGE STEGANOGRAPHY USING PYTHON

Tech Stack and platforms used to do the project:

Language: Python 3.8.4

Libraries: OpenCV-python 4.5.2

Numpy 1.20.3

Tkinter 8.6

IDE Used: Visual Studio

Platform used to deploy the project: Replit (connecting the GitHub repository)

2.HIDING ALGORITHM

DATA = AD

No. of **PIXEL REQ** = 3* Length(**DATA**)

= 3*Length(**AD**)

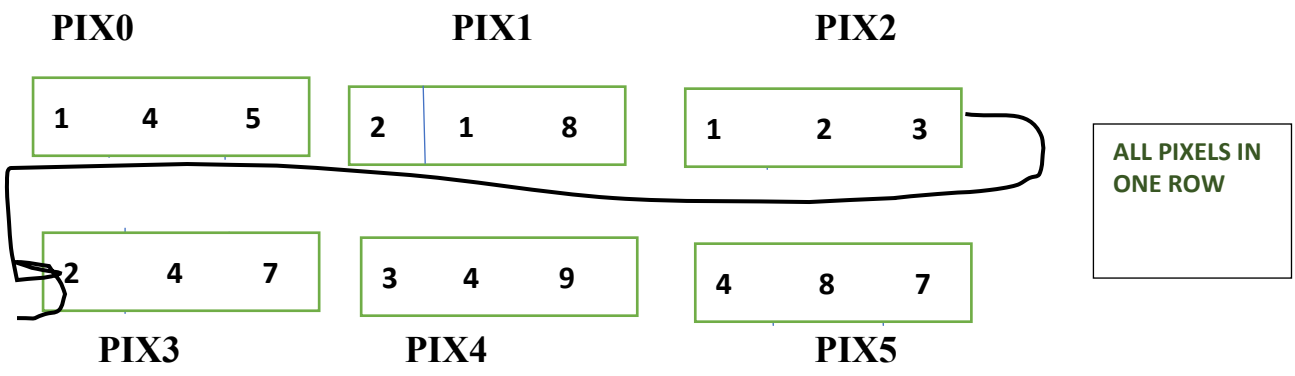
= 3*2

= 6

No. of **ROWS** = Math.ceil(**PIXEL REQ/Image width**)

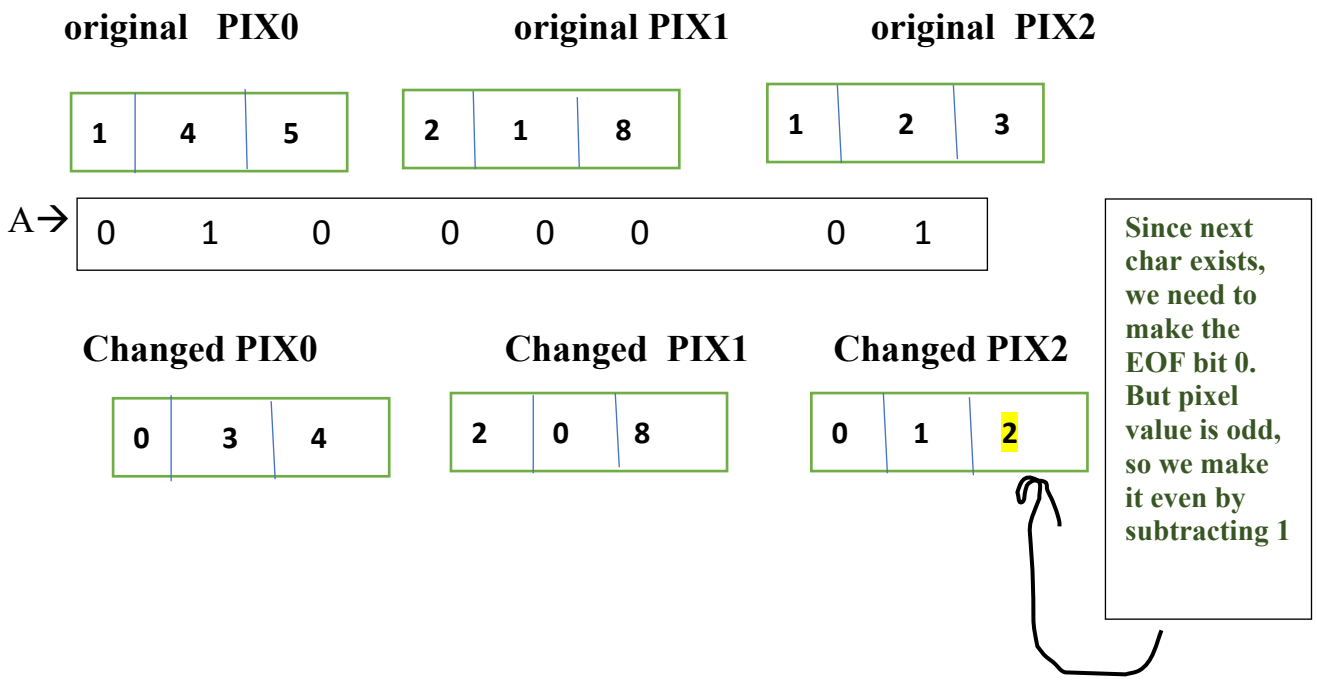
= Math.ceil(6/4160)

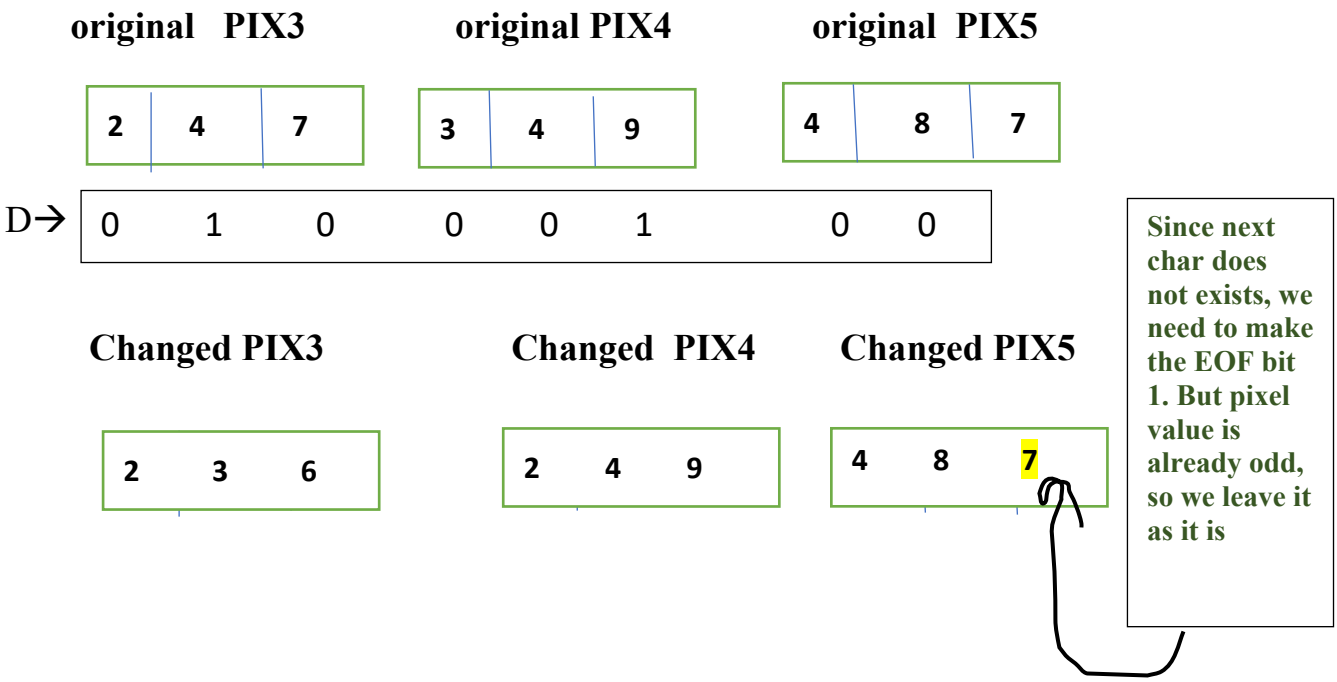
= Math.ceil(0.0014) = 1



- Check the number of pixels traversed. If the bit is 1 and the pixel value is an even number, make it an odd number by subtracting 1. Similarly, if the bit is 0 and the pixel value is an odd number, make it an even number by subtracting 1.
- Keep a count of the number of letters using the count variable.
- If the index is 7, check if the next character exists. If yes, mark the EOF bit as 0 (if the pixel value is odd make it even by subtracting 1, else leave it as it is) and continue. Else, mark as 1 (if the pixel value is even make it odd by subtracting 1 else leave it as it is) and end.

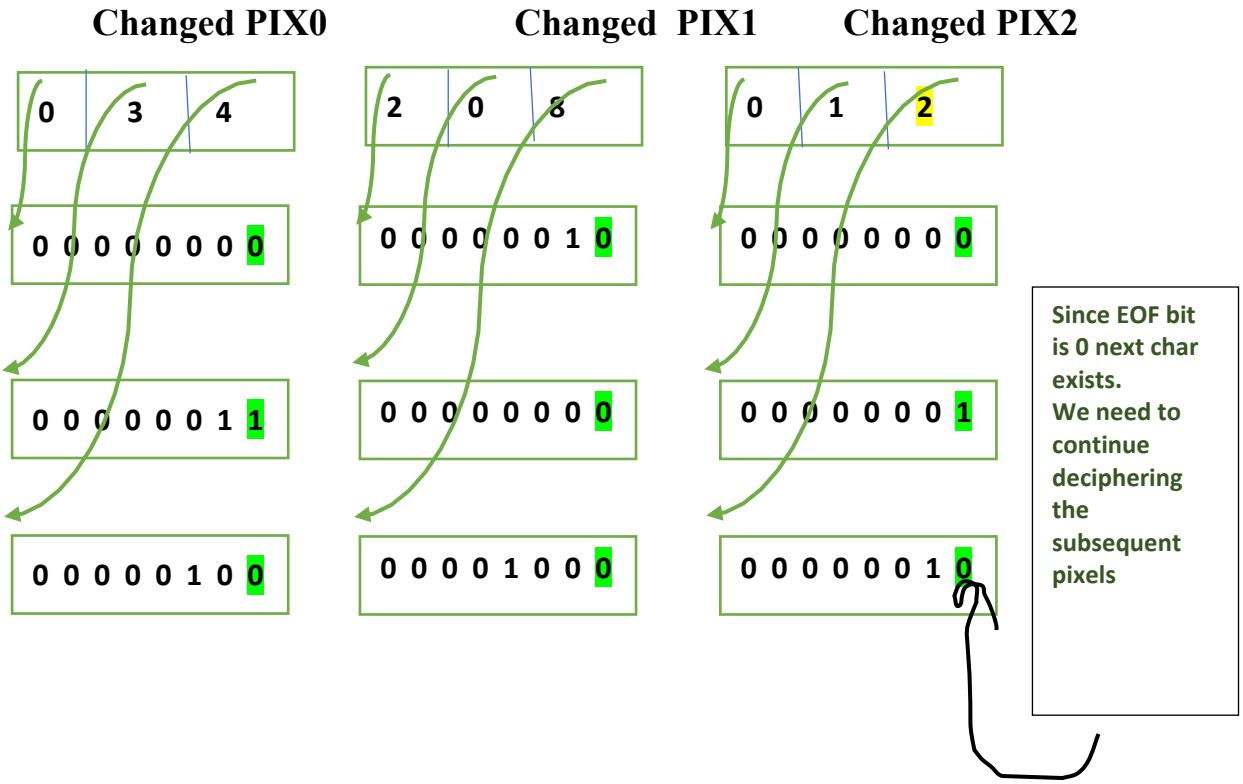
A→ 65 → 0 1 0 0 0 0 0 1 D→ 68 → 0 1 0 0 0 1 0 0





3.REVEALING ALGORITHM

1. Obtain the data from the stegnographed image by going through the "hide" algorithm.
    - Each of the 3 values in every pixel in every row has 1 bit of information, which is added into the data variable, using the for loop.
    - Check if the EOF character is reached.
        - If yes break from the for loop
        - Otherwise, continue.
    - The ASCII is stored serially in the data variable.



The last bit of all the binary form of pixel values is actually the binary form of the first character of the hidden data. The EOF is 0, which means there are more pixels to be deciphered.

Revealed data after first iteration: **0 1 0 0 0 0 0 1**

After 2nd iteration (concatenated)**: 0 1 0 0 0 0 0 1 0 1 0 0 0 1 0 0**

2. After obtaining the ASCII bits, bits are grouped into letters by making groups of 8.

   [0 1 0 0 0 0 0 1] [0 1 0 0 0 1 0 0]

3. The hidden message is then converted from binary to decimal form(ASCII Value) using the int(binary number,2) function.

[65] [68]

4. The ASCII value is converted into its original “character” form using chr(integer number)

[A] [D]

5. The letters are stored in the message variable, which is linked using the join command in python.
6. Finally, the proper message is shown on the screen.