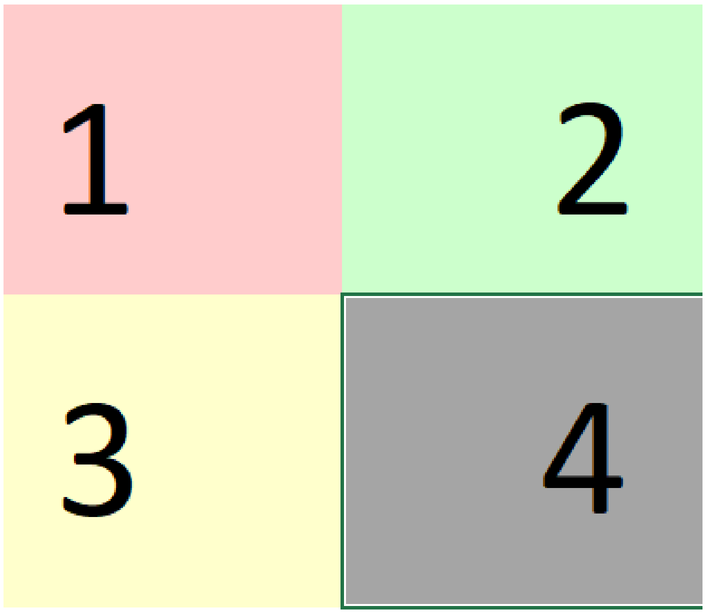
Standard JPEG compression methods are followed.

1.Block splitting

For the sake of simplicity, I will assume that we are working on a 32 x 32 colored image. If both sizes of x and y directions of the input image is not divided by 8, I simply add the smallest window possible to make it divisible by 8 without a residual. Each color represents a block.

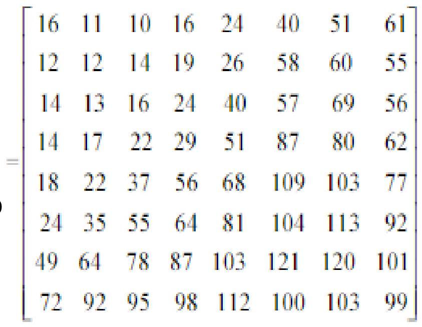


2. Discrete Cosine Transform

What I do in this step is to send each block to the DCT method in which all of the DCT's are calculated for every color channel, for this case red, green and blue. We had 4 blocks, so there are 12 8x8 double arrays after this step.

3. Quantization

Following matrix is used to quantize the DCT results by simply dividing each element by its counterparts from the quantization matrix. Decimal numbers are rounded to the nearest integer value and stored in an integer array. This step is irreversible and where we lose the information



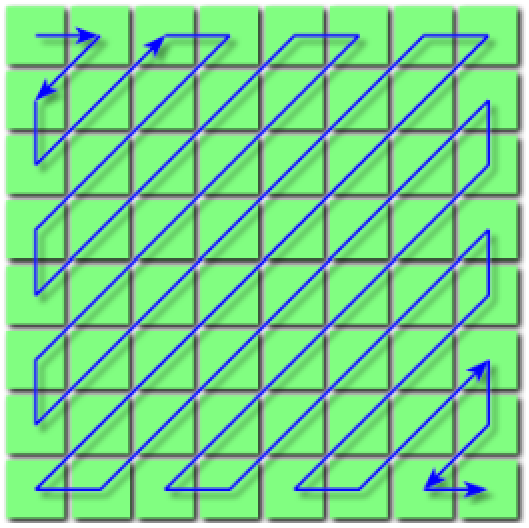
$$= \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

4. Entropy coding

This step consists of 4 major operations in my application.

4.1 Zigzag

In this step, I traverse the quantized matrix in a way to put high frequency values (where 0s are usually grouped to the right-hand side of the bottom) at the end of the 1D integer array. The zigzag sequence I used is below.



4.2 Huffman Encoding

Encoding the sequence process consists of two main operations. First, I create a Map for mapping each pixel value to how many times it is repeated in the sequence. Most frequent pixel values are put high (closer to the root) and least frequent values are put away from the root. Encoding method returns a string which consists of 1s and 0s. Thanks to the way we construct this string, it occupies way less memory than an integer array of 64 elements.

4.3 Huffman Decoding

Decoding process takes the encoded string and returns us the integer array

4.4 Re-Zigzag

All I do in this step is to re-create the 2D array and to put the values back

5. De-Quantization

I used the quantization matrix above to multiple each value corresponding to the indices.

6. Inverse discrete cosine transform

IDCT is applied to the re-quantized matrix to get to real pixel values back

7. Constructing the new image

Each block is put in its place to create the result image