Lab 10: Compression: Run-Length Encoding

Mingqin Dai

I.Motivation

The purpose of digital image compression is to store or transmit images using fewer bits than in the original image file. Compression is possible because of correlations between pixel values and/or between color components. If the original image can be recovered exactly, the compression technique is said to be lossless; otherwise it is lossy. In this lab, we focusing on lossless compression techniques (lossless coding method) on grayscale or binary images. We are doing this lab to do simulation experiments of four lossless coding method, which are "straightforward-brute force" coding, run-length coding, bit-plane coding and predictive coding, and evaluate the effects of them. Compression techniques can solve problem including high rate of digital image data transfer or data rate and large total amount of digital storage required or data capacity. Compression techniques help saving space (storage) and time.

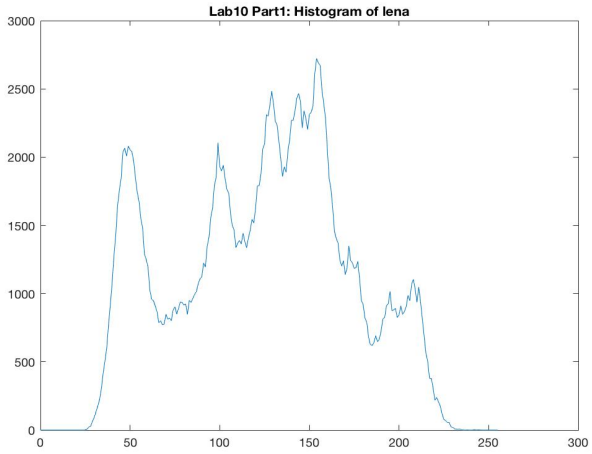
II.Method

Part 1: Entropy of Images

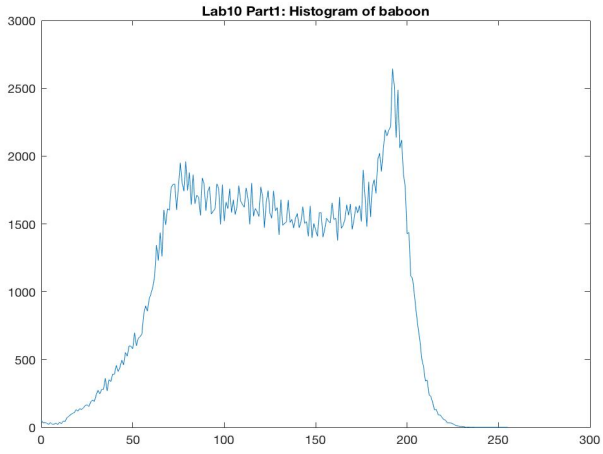
We calculate entropy of each image in this part. In order to do this, we first compute the histogram of given image. Then find the probabilities $P(i)$ from the histogram by normalization, which simply divide each element of the histogram array by the total number of pixels of the image. Finally, we using the function $H = -\sum_{i=0}^{255} P(i) log_2(P(i))$ to calculate the entropy of images.

a. Histograms

1.Histogram of lena



2.Histogram of baboon



b. Output Characteristics

1.Lena

Entropy of lena = **7.44551**.

The total number bits requires for lena = $\mathbf{1.95179\times10^6}$ bits.

The number of bits per pixel requires of lena = **7.44551**.

2.Baboon

Entropy of baboon = **7.47443**.

The total number bits requires for baboon = $\mathbf{1.95938\times10^6}$ bits.

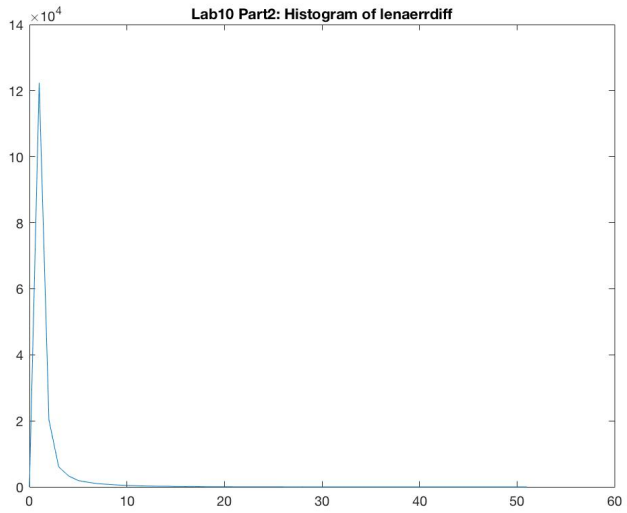
The number of bits per pixel requires of baboon = **7.47443**.

Part2: Run Length Coding of Binary Images

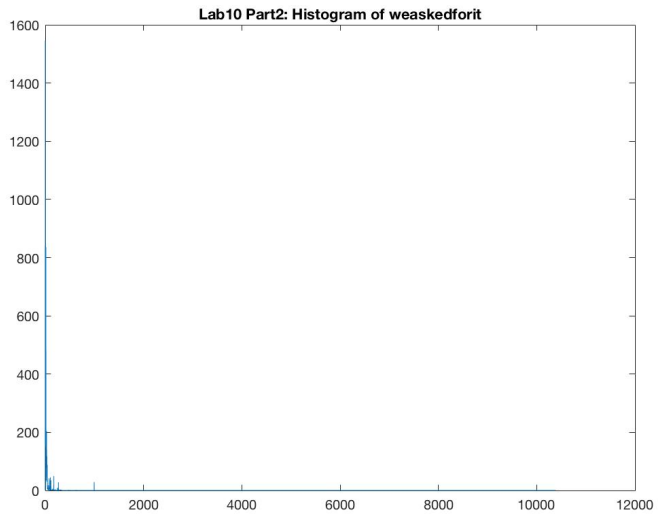
In this lab, we applying run-length coding(RLC) to bi-level images lenaerrdiff.png and weaskedforit.png. In order to complete compression using this method, we first scan the whole image in the so-called "raster-snake" order. Second, we find the "run-length" of the binary image and find the histogram that corresponds to the "run-lengths" we found. Third we find the probabilities $P(i)$ from the histogram by normalization, which simply divide each element of the histogram array by the total number of runs. Finally, we find the entropy $H = -\sum_{i=0}^{255} P(i)log_2\big(P(i)\big)$ from this histogram, which correspond to the bit rat that is going to spent to code each run on average. And we also find the total number of runs, the total number bits = $H\times total\ number\ of\ runs$, the average number of bits that is going to be spent per pixel = total number of bits ÷ number of pixels.

a. Histograms corresponds to run-length

1.Histogram of lenaerrdiff



2.Histogram of weaskedforit



b. Output Characteristics

1. lenaerrdiff

Run-Length Entropy of lenaerrdiff = **1.35375**.

The total number of runs of lenaerrdiff is **159928**.

The total number of bits requires of lenaerrdiff is **216503 bits**.

The number of bits per pixel requires of lenaerrdiff is **0.825894 bits**.

2.Histogram of weaskedforit

Run-Length Entropy of weaskedforit = **5.17333**.

The total number of runs of weaskedforit is **10224**.

The total number of bits requires of weaskedforit is **52892.2 bits**.

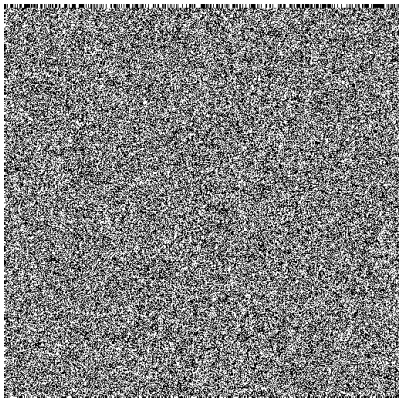
The number of bits per pixel requires of weaskedforit is **0.201768 bits**.

Part3: Run Length Coding of Image Bit Planes

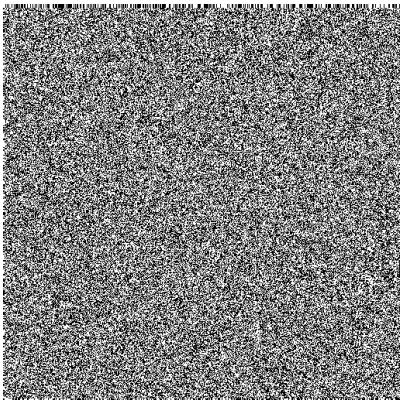
In this part, we first separate a given grayscale image into eight different images, which are all bi-level and are called the bitplanes of the original image. Then we applying run-length coding(RLC) to the eight bi-level images (bitplanes). (The run-length coding method is exactly same as Part2.) In the end, we find the run-length histogram, entropy $H = -\sum_{i=0}^{255} P(i) log_2\big(P(i)\big)$, total number of runs, the total number bits = $H \times total\ number\ of\ runs$, the average number of bits that is going to be spent per pixel = total number of bits ÷ number of pixels of the eight bitplanes.

a.bit plane images

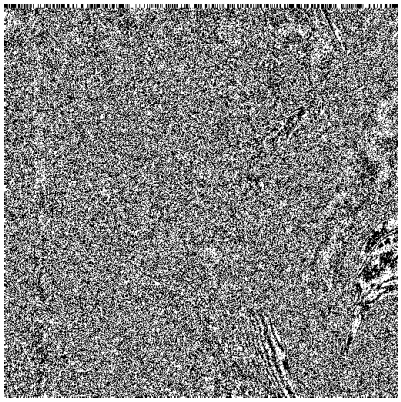
1.lena bit plane 0 (LSB)



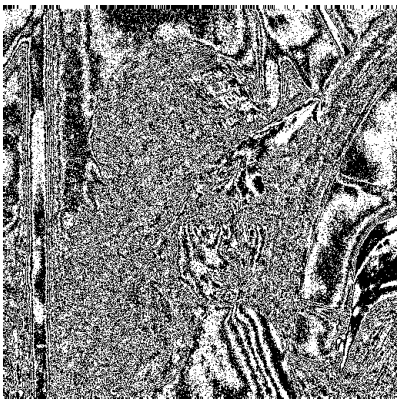
2.lena bit plane 1



3.lena bit plane 2



4.lena bit plane 3



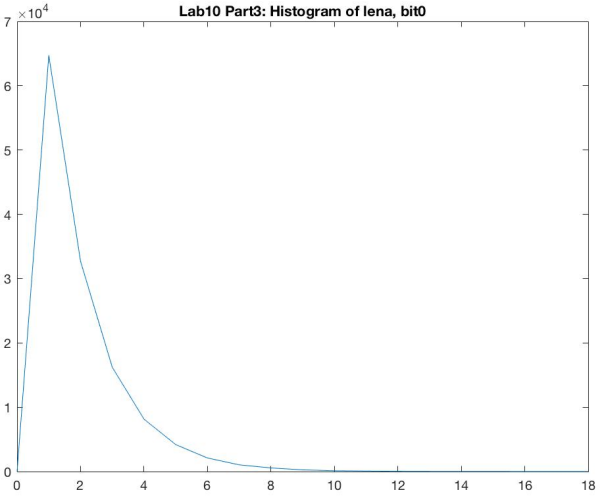
5.lena bit plane 4

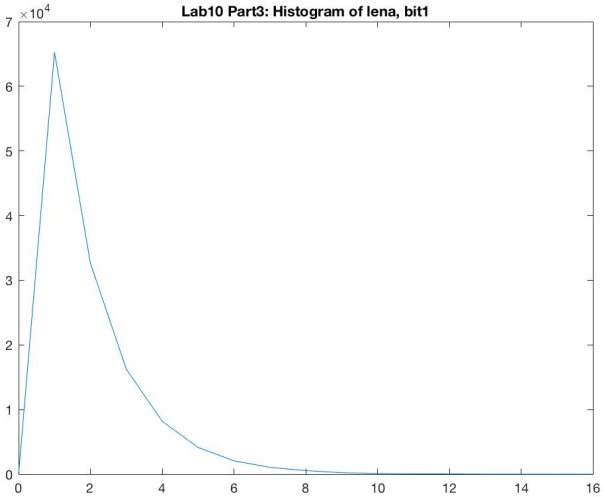
6.lena bit plane 5

7.lena bit plane 6 *8.lena bit plane 7(MSB)*

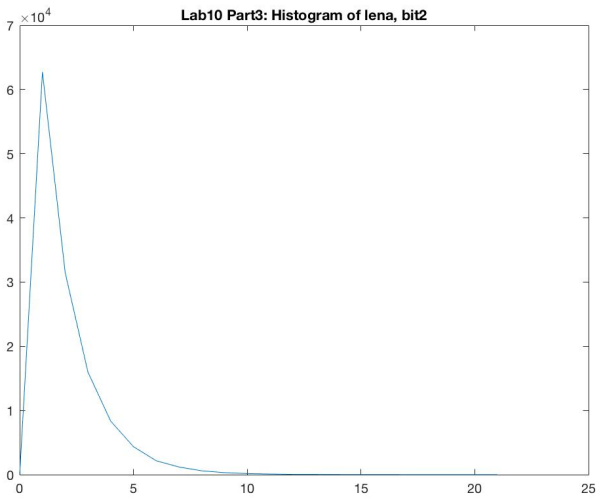
b.Histograms of bit plane images corresponding to run-length

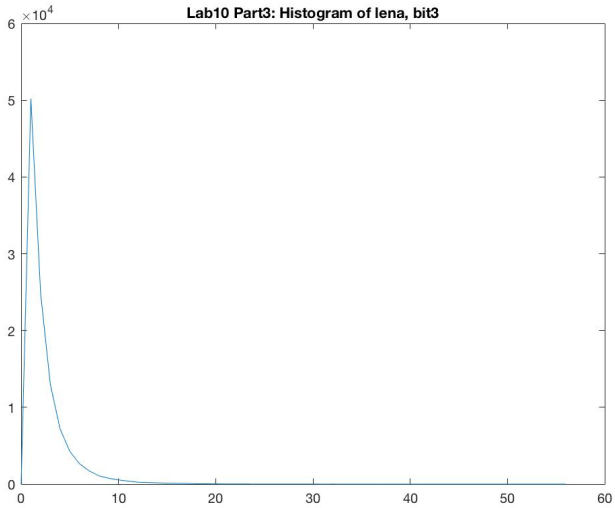
1. Histogram of lena bit plane 0 (LSB) *2. Histogram of lena bit plane 1*



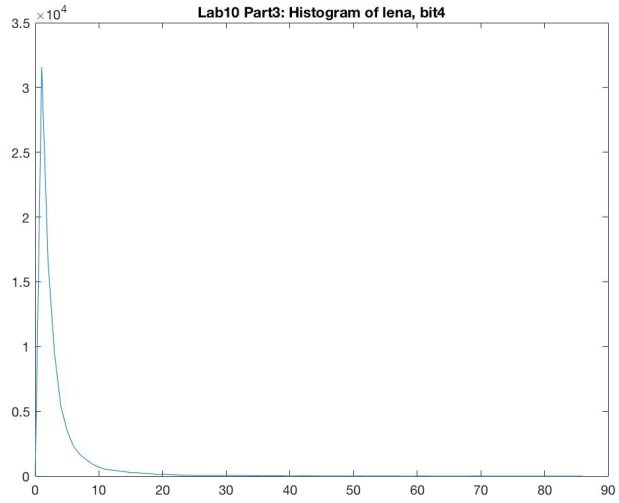


3. Histogram of lena bit plane 2 *4. Histogram of lena bit plane 3*

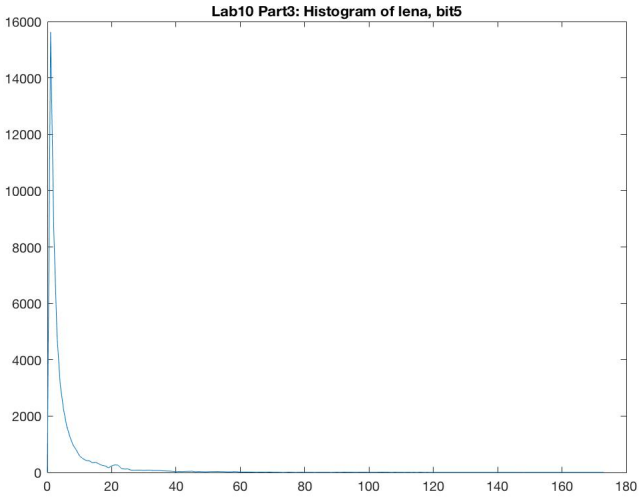




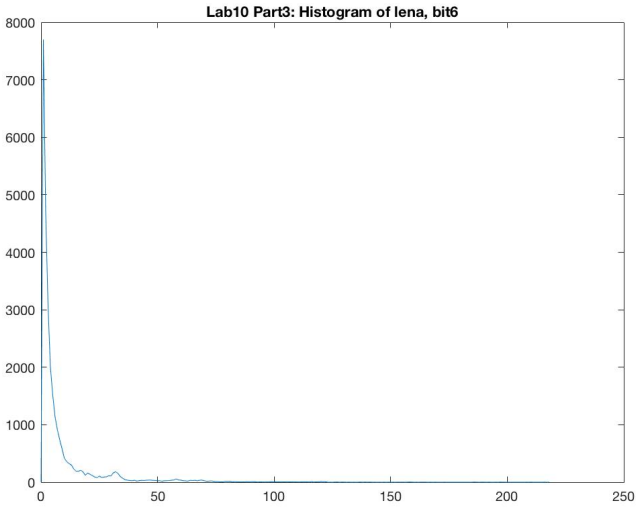
5. Histogram of lena bit plane 4



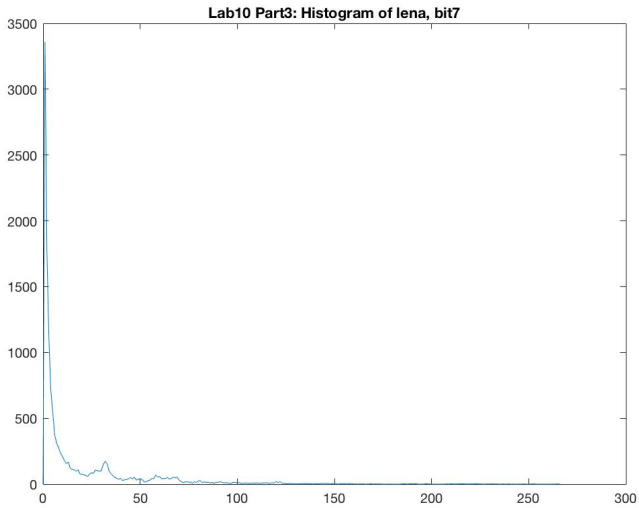
6. Histogram of lena bit plane 5



7. Histogram of lena bit plane 6



8. Histogram of lena bit plane 7 (MSB)



c. Output Characteristics

1. bit plane 0

Run-Length Entropy of bit plane 0 = **2.01304.**

The total number of runs of bit plane 0 is **130213**.

The total number of bits requires of bit plane 0 is **262124 bits**.

The number of bits per pixel requires of bit plane 0 is **0.999924 bits**.

2. bit plane 1

Run-Length Entropy of bit plane 1 = **2.00557**.

The total number of runs of bit plane 1 is **130700**.

The total number of bits requires of bit plane 1 is **262129 bits**.

The number of bits per pixel requires of bit plane 1 is **0.999941 bits**.

3. bit plane 2

Run-Length Entropy of bit plane 2 = **2.05428**.

The total number of runs of bit plane 2 is **127513**.

The total number of bits requires of bit plane 2 is **261948 bits**.

The number of bits per pixel requires of bit plane 2 is **0.999252 bits**.

4. bit plane 3

Run-Length Entropy of bit plane 3 = **2.34021**.

The total number of runs of bit plane 3 is **107857**.

The total number of bits requires of bit plane 3 is **252408 bits**.

The number of bits per pixel requires of bit plane 3 is **0.962861 bits**.

5. bit plane 4

Run-Length Entropy of bit plane 4 = **2.81322**.

The total number of runs of bit plane 4 is **77499**.

The total number of bits requires of bit plane 4 is **218022 bits**.

The number of bits per pixel requires of bit plane 4 is **0.831688 bits**.

6. bit plane 5

Run-Length Entropy of bit plane 5 = **3.49383**.

The total number of runs of bit plane 5 is **45668**.

The total number of bits requires of bit plane 5 is **159556 bits**.

The number of bits per pixel requires of bit plane 5 is **0.608659 bits**.

7. bit plane 6

Run-Length Entropy of bit plane 6 = **4.08881**.

The total number of runs of bit plane 6 is **28387**.

The total number of bits requires of bit plane 6 is **116069 bits**.

The number of bits per pixel requires of bit plane 6 is **0.442768 bits**.

8. bit plane 7

Run-Length Entropy of bit plane 7 = **5.0021**.

The total number of runs of bit plane 7 is **13926**.

The total number of bits requires of bit plane 7 is **69659.3 bits**.

The number of bits per pixel requires of bit plane 7 is **0.265729 bits**.

9. total number of bits per pixel of lena

the total number of bits per pixel for all the bit plane images of lena combined is **6.080822**.

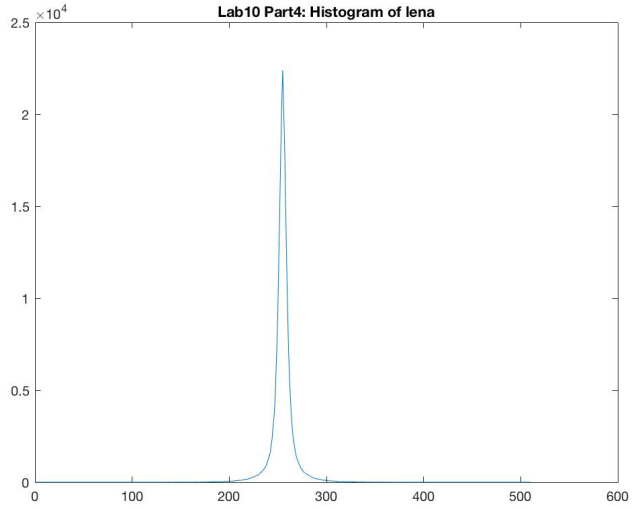
Part 4: Predictive Coding: "Raster Snakes"

In this part, we applying predictive coding to images same as in part 1 (lena.png; baboon.png). First, we scan in the image pixel's intensity in "raster snake order". Second, we predicting the intensity value of pixel X by using the previous pixel intensity value a. Third, we find the differences between the original pixel and its predicted value $X - a$, which is the prediction errors. There are $512 \times 512 - 1$ difference values in total. We can reconstruct the image using the first pixel intensity value ad prediction errors. In the end, we find the histogram of the difference pixel values, which ranges from -255 to 255 (size of 511 in total). And we find the entropy by dividing the histogram using the same method as Part 1. Also, we find the total number of bits and average number of bits that is going to be spent per pixel.

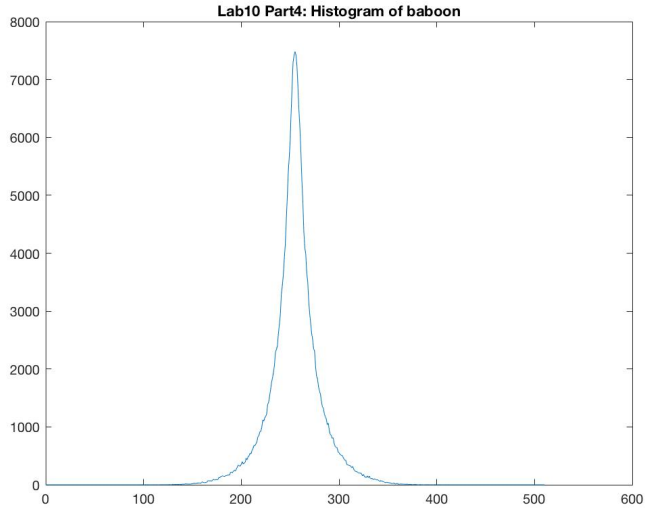
a. Histograms

For the histograms, it has size of 511 in total, range from -255 to 255. I plot the histogram from o to 510 corresponding to -255 to 255.

1. Histogram of lena



2. Histogram of baboon



b. Output Characteristics

1.Lena

Prediction Error Entropy of lena = **5.05501**.

Prediction Error Entropy of lena is $\mathbf{1.32514 \times 10^6}$ **bits**.

The number of bits per pixel requires of lena = **5.05501**.

2.Baboon

Prediction Error Entropy of baboon = **6.54799**.

Prediction Error Entropy of baboon is $\mathbf{1.71651 \times 10^6}$ **bits**.

The number of bits per pixel requires of baboon = **6.54799**.

Part 5: Discussion (Answering questions)

Question 1 (histogram plots and required characteristics) is answering in the above parts (Part1 to Part4). ***(Q1)***

According to the histograms I plot, Lena.png has a more sharply peaked histogram than Baboon.png. And according to the entropy I found, Lena.png gives a lower entropy compared with entropy of Baboon.png. We can deduce a noticeable correlation between entropy and the shape of the histogram from the above information that the entropy for a more sharply peaked histogram would be lower. I think my conclusion is true in general, because more sharply peaked histogram indicating less disorder or uncertainty of the grayscales, which is actually lower entropy. In the extreme case of a histogram concentrated at a single gray level, the entropy would be 0 bpp. ***(Q2)***

According the result of part 2, comparing the number of bits per pixel requires, lenaerrdiff.png (0.825894 bits) has apparently more bits per pixel than weaskedforit.png (0.201768 bits). Comparing the run length entropies, the run length entropy pf lenaerrdiff.png is lower than that of weaskedforit. This is because lenaerrdiff has less run length values (size of histogram corresponding to run length is 52) than weaskedforit.png (size is 10384), lenaerrdiff is more sharply peaked. We only consider bi-level images in part 2. In general, smoother images, which refers to images do not have wildly different values between adjacent pixels, could compress the best. Or in other word, images contain longer runs of 0s and 1s could be compressed more efficiently. If we considering entropies, the result gives that images with higher run length entropy could compress the best. We first analyze why this type of images are easier to compress in terms of the general shape of the image. Smoother images have longer runs of 0s and 1s. The repeated consecutive values could be represented by the repeated times of the same value. So with each run length all represented by the repeated times (, which means every different runs all compressed to same number of bits), longer runs refers to higher compression efficiency. Thus smoother images could compress the best. Second, we analyze this question in terms of histograms. Considering histograms corresponding to run length, in general larger size of histogram refers to smaller total number of runs and larger run length entropy (Assuming the size of image is 512 times 512). The total number of runs could be extremely large, especially compared with the run length entropy. Hence the total number of bits requires is dominant by the total number of runs, which means larger total number of runs gives higher compression efficiency. Thus images with apparently larger size of run length histogram (or larger entropy of run length histogram) is easier to compress. (I don't analyze this corresponding to the grayscale histogram because for bi-level images, the histogram only gives the number of 1s and 0s instead of the correlation between adjacent pixels.) ***(Q3)***

Bit plane 7, which is the most significant bit plane, could be compressed most efficient. Whereas bit plane 0, which refers to the least significant bit plane, has the least compression efficiency. This is because MSB is the smoothest bit plane among the eight bit planes, which means it contain longest runs of 0s and 1s and lowest number of runs. For the same reason explained in Q3, MSB could be compressed most efficient (0.265729 bits/pixel). On the contrary, LSB is the noisiest bit plane among the eight bit planes, which means it mostly contains shorter run length (the longest run length is 18). With mostly shorter run length and large number of runs, it almost incompressible (0.999924 bits/pixel). (Same reason as in Q3.) **(Q4)**

According the result, the predictive coding gives both lower entropy and number of bits per pixel for both lena.png and baboon.png compared with "straightforward-brute force" coding. The predictive coding using the property of image that successive pixels on each row of an image are highly correlated. Hence the value of given pixel X may be predicted using the previous pixel a with the first pixel of the image need to be encoded individually. Based on the first pixel and the prediction errors X-a, the decoder can exactly reconstruct the entire image. Prediction errors is integer-valued and usually small. They are large near edges, so edges and fine details are highly visible in the error image. Hence the histogram for the prediction errors should be more sharply peaked around difference error equals to zero. (This result could also be derived by comparing the histogram of original image (Part 1) and histogram of difference errors (Part 4).) Entropy of predictive coding is thus lower (5 bits versus 7 bits). With entropy corresponding to the lowest possible rate for a VLC, the predictive coding with obviously lower entropy performs much better than "straightforward-brute force" coding. **(Q5)**

I consider this question first in terms of RLC of a grayscale image. For an example string *ABBBBBBBBBCDEEEEF*, which takes up 1 bit/pixel. Using RLC compression, we get *A *8B C D *4E F,* which takes up 10/17 bit/pixel. The file has been compressed effectively. However, with this representation method, RLC is only effective if there are sequences of 4 or more repeating characters because three characters are used to conduct RLC. So coding two repeating characters would even lead to an increase in file size. After that, I consider the question in terms of bi-level image, 1111100000111100000. Using RLC compression, we get 1*50*61*45*0, which compress the file effectively. Considering the problem that happened on grayscale image above that expand the file size, RLC of bi-level images could also result in file size expansion using same representation method. However, with the condition that the image is consisted of 1s and 0s, we don't actually need 1* or 0* (repetitive character itself) in front of the run length, because the value varies for every runs and the value is binary. So what we only need is the value of the first pixel of the image and the run

length (number of repeated characters) for the whole image. Thus it can be represented as 15645 or the quadruple (5,6,4,5) (this is the compactly representation given on the note). Considering the extreme situation that all adjacent pixels have different values, if we using the first representation method (15645 one), the resulting string is 1 character longer than the original string. Thus it expands the image in terms of bits per pixel really slightly. If we using the second method ((5,6,4,5) one), considering the extreme situation, it will result in a two times expansion of the image in terms of bits per pixel. **(Q6)**

Comparing the total of my results from part 3 with my result for the lena.png in part1, it is apparent that bit-plane run length encoding which encoding each bit plane separately takes less bits than Huffman coding done on the 256-level image as in Part 1. For bit-plane RLC, though LSB is almost incompressible, MSB with long runs 0s and 1s would be compressed effectively. In general, the bit-plane run length encoding takes less bits than Huffman coding. However, if we consider the extreme situation that a chessboard-form image with 1x1 blocks consisting with grayscale 170 and 85, it gives different answer. Using Huffman coding, the entropy $H = -\sum_{i=0}^{255} P(i) log_2\big(P(i)\big) = 1$. While 170 and 85 refers to 10101010 and 01010101, the 8 bit planes are all consisted of 10101010… or 01010101…. From the result I got from Q6, this may cause expansion in image file size of all eight bit planes (or at least same with the original size). Hence result in expansion of bits per pixel of the original lena.png. Thus in this case, the run-length encoding each bit plane separately takes as many or more bits than Huffman coding done on the 256-level image**. (Q7)**

III.Results

My results are as follows. First, images with lower entropies could be compressed more efficiently. Second, run-length coding is more effective for smoother images. Third, considering bit-plane run-length coding, most significant bit-plane could be compressed most efficiently while least significant bit-plane is almost incompressible. Forth, predictive coding is more effective than "straightforward-brute force" coding, especially for smoother images. I learned that the entropy for a more sharply peaked histogram would be lower, which means smoother images could be compressed more efficiently. And for run-length coding for binary images, images contain long runs of 0s and 1s could be easily compressed. Also, predictive coding for smooth images is considerably effective compared with "straightforward-brute force" coding, because it considers the nature of most images that successive pixels on each row of an image are highly correlated. Compressing techniques above help saving space (storage) and time in most case.