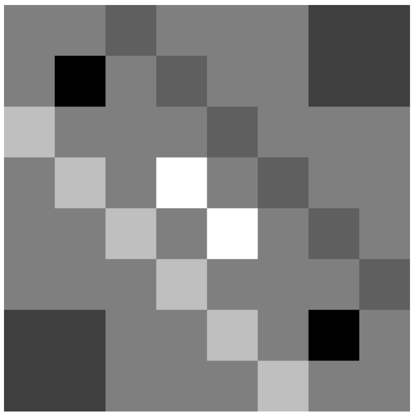
# CS4481 Assignment #2

## Test Image #1 – test_square.raw.pgm



test_square.raw.pgm – 257kb

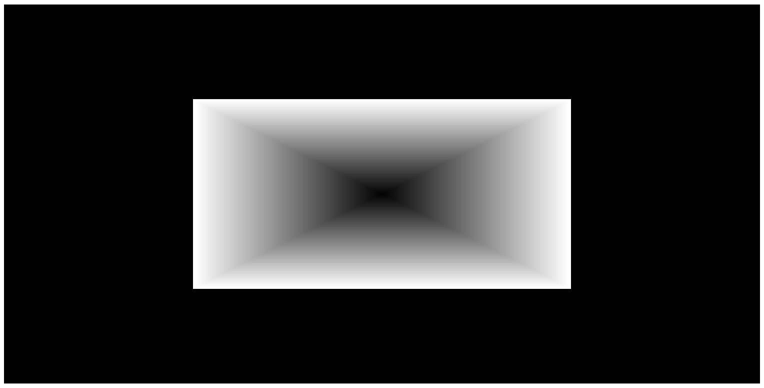
test_square.comp – 59kb

test_square.decomp.pgm – 257kb

MAE(test_square.raw.pgm, test_square.decomp.pgm) = 0

Compression of test_square.raw.pgm is excellent because there are few shades used and one shade that is used the majority of the time. Since one shade appears very frequently, the Huffman code assigned to it is shorter than the others, providing excellent compression.

## Test Image #2 – rectangle_2.raw.pgm



rectangle_2.raw.pgm – 129kb

rectangle.comp – 44kb

rectangle_2.decomp.pgm – 129kb

MAE(rectangle_2.raw.pgm, rectangle_2.decomp.pgm) = 0

Compression of rectangle_2.raw.pgm is successful because a single color (black) takes up just over half of all pixels. To provide optimal compression, the color used most frequently is assigned the shortest code and this is why the image is compressed very well.

## TEST IMAGE #3 – BOATS.RAW.PGM

boats.raw.pgm – 241kb

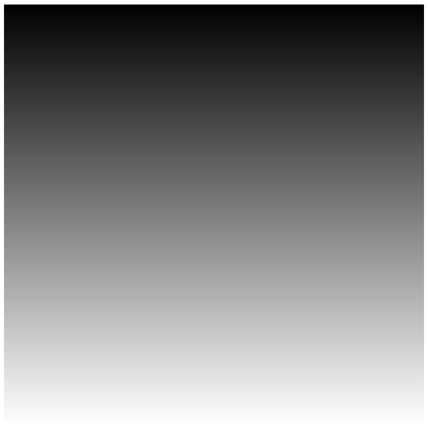
boats.comp – 218kb

boats.decomp.pgm – 241kb

MAE(boats.raw.pgm, boats.decomp.pgm) = 0

Compressing boats.raw.pgm is not an extremely impressive example of what Huffman encoding is capable of, but compression is still successful to a relatively small degree. There is a wide range of shades used, with higher levels of gray (lighter shades) being used more frequently they are assigned shorter Huffman codes. Interestingly, there appears to be only one pixel in the bottom right corner that has the value 0 (black), and in testing, the pixel with the value 0 has a significantly longer Huffman code than any other pixel in the image. This is expected because symbols with lower probabilities should have longer codes, while symbols with the most probability should be assigned the shortest codes to make for better compression.

## TEST IMAGE #4 – SMOOTH.RAW.PGM



smooth.raw.pgm – 65kb

smooth.comp – 67kb

smooth.decomp.pgm – 65kb

MAE(smooth.raw.pgm, smooth.decomp.pgm) = 0

Compressing smooth.raw.pgm is a special case where every color 0-255 is used, and each color is equally likely. This results in each Huffman code being 8 bits long, the same number of bits that would be used in the original pgm encoding. The compressed file would not only store each pixel at the original size of 8 bits long, but would also have to store the “header” of the Huffman encoding (length, width, number of nodes, all Huffman nodes, etc), which is larger than a pgm header. With the extra overhead of the Huffman encoding, this particular case does not see any benefit to compressing with Huffman codes.

## Test Image #5 – flat.raw.pgm

flat.raw.pgm – 238kb

flat.comp – 30kb

flat.decomp.pgm – 238kb

MAE(flat.raw.pgm, flat.decomp.pgm) = 0

Compressing the file flat.raw.pgm was the best case scenario for Huffman encoding because there is only one color. This means the only Huffman code needed is one bit long, providing the best compression out of all the sample pgm images.