Homework Assignment #8

Due Date: Friday, 6 December, 2013 at 11:59pm (MDT)

What to Submit

Please submit **ONE** appropriately-named **pdf file** via Learning Suite containing your solutions to both the WRITTEN and the PROGRAMMING exercises (include the necessary plots and all code that you've written to complete the programming exercises).

WRITTEN EXERCISES

1) Use the linked page of English letter frequencies to determine the entropy of this set. What would the entropy be if each letter occurred with the same frequency (probability)?

   (The entropy for the English language is much lower once you include the often-occurring space, the infrequently occurring punctuation, and the predictability of multiple-character sequences.)

   Note: You don't have to do all of these calculations by hand. You are welcome to use a calculator, spreadsheet, or program **of your own design and implementation**.

PROGRAMMING EXERCISES

For this assignment, implement a simple compression algorithm similar to that used in the lossless version of JPEG (and at one point as part of the lossy version as well).

The Unix `compact` program uses Huffman coding to compress a file. As we have discussed (or will discuss) in class, the effectiveness of entropy encoders can be improved by using a predictive encoder to decrease the entropy. (If we haven't covered this yet, read Section 8.4.4 of your textbook.)

1) Select one of the images that we've been using in class, copy it to your directory, and run the Unix `compact` command on it. Record the size of the compressed file.
2) Write a program that uses predictive coding to reduce the entropy of an image. The program should have a single greyscale image as input and product a single greyscale image as output. Use a variation of the lossless JPEG standard where each pixel is predicted as the weighted average of the adjacent 4 pixels already encountered in raster order (three above and one to the left). That is, try to predict the value of a pixel ***a*** based on preceding pixels, *x*, to the left as well as above ***a***:

x	*x*	*x*
x	***a***	

Then calculate and encode the residual (difference between predicted and actual value).

Note: even though the residual may have both positive and negative values, [-256,256], it can still be encoded in one byte since the error for a prediction of P is in the range [-P,256-P].

3) Display and compare the histograms for the image before and after predictive encoding. Specifically, how does the dynamic range compare? Which is a better candidate for entropy encoding, the original or the residual? Why?
4) Apply the `compact` command to the predictively-encoded image. Record the size of your newly compressed file.
5) Now use `compact -d` and your own predictive decoder to decompress the file.
6) Display the image to make sure your compression/decompression code works. (Hint: you may want to do this step earlier in order to test your predictive encoder/decoder.)
7) Once you've verified that your code works, test it on at least five images. Record for each the size of the original image, the size of the compacted file, and the size of your predictively-encoded/compacted file.
8) For additional credit (up to 10% of this homework), experiment with different or even multiple predictive contexts. The better the compression rate, the more credit you'll earn.

Note: the `compact` command is not installed on our current Linux configurations. Here is a compiled executable for it. The source is also available and may work on Windows systems. You may also use another Huffman coder if you prefer, but make sure it's doing only Huffman coding.

Caution: some image I/O programs or libraries scale the results for display when writing a file. If for some reason your output image does not include both 0 and 255 as pixel values, the residual values might be (incorrectly) scaled to this range. This won't in general be a problem, but if you encounter it, just change the first pixel (the one you encode absolutely instead of relatively) to 255. If necessary, you may also have to set the residual on the last pixel to 0, but that's usually not required.

Programming Exercise Write-up

Please prepare a written write-up (submitted as a PDF) which includes the following:

- All code that you wrote for the exercises (programming and written if you coded that).
- The relevant plots and answers to the programming question.
- A brief explanation of what you did, any challenges you encountered, things that were difficult, unclear, etc.
- How long did the different parts (written, programming) of the assignment take you?