Homework #2

Due Date: 11:59pm Saturday 29 March 2014

In this homework you will explore the concept of entropy, and write a C++ program that can compress and decompress images using the Huffman Coding algorithm. Your program will read the image on `stdin` and outputs the resulting image on `stdout`, and get in the required operation on the command line.

Please present a report containing your answers as well as a zip file containing all your code.

1. [3 points] Implement the function `compute_entropy` to compute the entropy for each of the included PPM files. The symbols are the grayscale values from 0 to 255 (one byte each). Estimate the probability by the relative frequency of occurrence of each value i.e. $P(v) = count(v)/N$ where *count*(*v*) is the number of pixels with value *v* and *N* is the total number of pixels in the image and

   [Note: any grayscale value that is not in the image is ignored when computing the entropy since its probability will be zero and $\log_2 0$ is infinity.]

   [Hint: Pay special attention to the files `msgx.pgm`. You can compute the entropy by hand to verify your code is working correctly.]

2. [4 points] Implement the function `build_tree` to build a Huffman Tree that takes in symbol probabilities computed in the previous step. Print the computed Huffman Tree and the codes for the symbols for the images `msg1.pgm`, `msg2.pgm` and `msg3.pgm`.

3. [4 points] Implement the function `encode` to encode an input message and output an array of codewords. You don't need to worry about bit compaction for the this homework, but you need to keep track of how many bits are in the encoded image.

   Encode the four images that were shown in class (sena, sensin, earth, omaha) and compare the numbers of bytes you get for the compressed image with the numbers given in class. Comment on your results.

4. [3 points] Implement a simple *decorrelation* method to try to reduce the number of bytes needed for the compression process. In particular, instead of encoding the given grayscale values directly, encode the different between each pixel and the one on its left. So, given an input image, compute the *difference* image by subtracting from each pixel the value of its *left* neighbor, and then use that image to build the tree and encode. Note that for the first column there is no left neighbor, so just use the pixel values.

   Encode the four images that were shown in class (sena, sensin, earth, omaha) and compare the numbers of bytes you get for the compressed image with the numbers given in class. Comment on your results.

Command Line

You need to modify the main file `hw02.cpp` to include the required functionality. Your program should be named hw02, and should be called as follows:

- To compute entropy for an image:

  ```
  ./hw02 -entropy < input.ppm
  ```

  this will output the entropy of the image `input.ppm` , for example

  ```
  ./hw02 -entropy < msg1.pgm
  ```

- To build a Huffman Tree from an image and prints it:

  ```
  ./hw02 -tree < input.ppm
  ```

  where the input image is called `input.ppm` and the output is written to `stdout`. For example, to compute the Huffman Tree on `msg1.pgm`, you could run:

  ```
  ./hw02 -tree < msg1.pgm
  ```

- To encode an image:

  ```
  ./hw02 -encode < input.ppm
  ```

  where the input image is called `input.ppm` and the number of bytes used in the encoding is written to `stdout`. For example, to encode `sena.pgm`, you could run:

  ```
  ./hw02 -encode < sena.pgm
  ```

- To encode an image by computing the difference first:

  ```
  ./hw02 -encode -diff < input.ppm
  ```

  where the input image is called `input.ppm` and the number of bytes used in the encoding is written to `stdout`. For example, to encode `sena.pgm`, you could run:

  ```
  ./hw02 -encode -diff < sena.pgm
  ```

Instructions

- All code should be implemented in C++ under Linux.
- Please submit your homework in one zip file named as follows:

CMPN206.HW##.FirstName.LastName.zip, so for example if your name is Mohamed Aly and this is homework #1, then the file name should be *CMPN206.HW01.Mohamed.Aly.zip*.

- Please include all your code and sample output in the zip file, with a README file to explain what you did. Failure to follow these instructions will cause deductions from your grade.
- You are allowed to discuss the problems among yourselves. However, **copying** any part of the code will result a grade of **ZERO**. No exceptions.

Grading

- 14 points: requirements above
- 1 point: submission instructions