15-122: Principles of Imperative Computation, Spring 2011
Assignment 6 Optional Challenge: Image Compression

Jason Koenig (`jrkoenig@andrew.cmu.edu`)

Out: Friday, March 25, 2011

1 Image Compression

So far we have only considered compressing textual data. We may also want to compress images for a similar reason as compressing text: to save disk space or bandwidth. There are many ways to compress images, some which lose information in a way that leaves the image largely similar to human perception, and others which recover the original information exactly. We can use Huffman codes to perform this "lossless," exact compression.

If try to compress the image values directly, we find that they are often relatively uniformly distributed, and thus would not compress significantly. However, we can make the observation that many images are relatively smooth: while individual pixels may have any value, the difference between successive pixels is often clustered around zero. These differences can often be compressed much more easily than the original pixels, because they are more concentrated and less uniform. This suggests the following algorithm (for an 8-bit, RGB image):

1. Write the top left pixel's value (`[0,256)`) as a bit sequence according to the Huffman code. (The frequency table contains enough entries to allow this value to be coded according to the same scheme as pixel differences. The distinction is the first value that is encoded is an absolute value, rather than a difference.)

2. For each subsequent pixel, going to the right along each row, and then wrapping around after the rightmost column, compute the difference between the previous and next pixel. Note that this means that the first pixel of each row is coded relative to the last pixel on the row before.

3. Using Huffman codes, turn this difference into a bit sequence, which is then written to the output.

4. Repeat with for each subsequent pixel until all have been encoded in the current channel. Note that there will be `width*height-1` codes, along with the seed value.

5. Repeat the above procedure for each of the red, green, and blue channels, in that order. Thus there will be three seed values, one for each channel.

The decoding algorithm works similarly but in reverse, using the same Huffman tree to find the difference, and thus inferring the value of the next pixel in the image. This algorithm works best when each of the three color channels is encoded separately (as the assumption of smoothness breaks down if the colors are interleaved). Thus the first decoded value of each channel is not a difference, but a seed value for that channel.

With this scheme, the provided image is about 60% of the uncompressed size, which is reasonably high compression given the simplicity of the algorithm. PNG is a popular lossless image compression format, and this algorithm often is comparable to PNG compression ratios for photographic images. (On computer generated imagery, PNG has special rules which allow it to get much better compression.)

2 The Prize

Your task is to write the code to decode the attach image data, and send an email to `jrkoenig@andrew.cmu.edu` with a short (1-2 sentence) description, including color information, of the subject of the compressed photo. The image is 600 pixels wide and 448 pixels high. The file `table.txt` contains the frequency distribution for differences (and the seed values). You should use all of the conventions given in HW6 when generating your Huffman tree so that your tree matches the one used for encoding. In addition, you should add the initial entries into the heap in increasing order (the file is in this order). Note that this file contains integers, not characters. You will need to modify the frequency table loading code accordingly. You should refer to the starter code of HW1 for handling images in C0. The file `encoding.txt` contains the bit sequence of the encoded imagery as a whitespace delimited series of ″0″'s and ″1″'s. You may find the `readfile.c0` file has useful functions for loading this data. You may find these files on the course website.

Any student who send me an accurate description of the image will win a prize!