Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science
Department of Mechanical Engineering

6.050J/2.110J **Information and Entropy** **Spring 2003**

Problem Set 2 Solutions

Solution to Problem 1: Is it Over-Compressed or is it Modern Art?

The following MATLAB code solves parts a to e:

```
% Perform initialization as indicated in the problem set.
load imdemos flower;
flower=double(flower);
colormap('gray');

% Since many figures will be produced by this script, we use meaningful labels.
set(gcf,'NumberTitle','off','Name','Flower'); imshow(flower,[0 255]);

% Implement the comrpession scheme detailed in the problem set.
encoded=blkproc(flower,[8 8],'dct2');
encoded(abs(encoded)<10)=0;
decoded=round(blkproc(encoded,[8 8],'idct2'));

% Provide the error value to check against the expected value from the set.
sprintf('With cutoff=10, the mean squared error is %.4f', ...  mean2((flower - decoded).^2))
```

The mean squared error you should have obtained is 10.6891. The next piece of code produces the graph of file size versus mean squared error.

```
% Initialize the vectors that will store the data for the graph.
x=[];
y=[];

% We need only encode the image once.  After that, since we will be steadily
% increasing the threshold, we need to reconvert again more because we will be
% simply zeroing-out more elements with each iteration through the for loop
% (there is no reason to recover all the original elements and start from scratch
% each time through the loop; we can progressively drop more and more data).
encoded=blkproc(flower,[8 8],'dct2');

% Now we begin to collect data for the graph.
for cutoff=0:4:100,
  encoded(abs(encoded)<cutoff)=0;
  decoded=round(blkproc(encoded,[8 8],'idct2'));

  % We will simply append to the vectors each time through this loop.
  x=[x,nnz(encoded)];
```

```
    y=[y,mean2((flower - decoded).^2)];

    % The next three lines can be commented out if they are not desired.  They
    % will produce a new window, label it, and print a representation of the
    % newly decoded image for each cutoff threshold.  This is for comparison
    % with the original image to answer the question in the set that asks at
    % which point the difference between the original image and the compressed
    % image becomes perceptible to the human eye.
    figure;
    set(gcf,'NumberTitle','off','Name',sprintf('cutoff=%d',cutoff));
    imshow(decoded,[0 255]);
  end

  % Now, plot the graph with a smooth curve and boxes around all the actual data points.
  figure;
  set(gcf,'NumberTitle','off','Name','Graph for Problem 1');
  plot(x,y,'s-')
  title('Comparison of File Size and Image Error');
  xlabel('Non-zero matrix values (number of bytes to store)');
  ylabel('Mean squared error');
```

You should have gotten something remotely resembling the graph in Figure 1. As you can see, there is a point where the MSE increases exponentially giving a quantitative value to the degradation of the reconstructed picture. Medical applications such as in X-rays tend to discourage the use of JPEG or similar lossy compression algorithms for saving images due to chances of distortion leading to an incorrect diagnosis.

The largest byte size for which I could detect the difference between the original image and the reconstructed image by eye was 1,785 bytes, which corresponds to a cutoff of 24 and a mean square error of 42.2132. This is rather remarkable because the image started out over 16kb long, which means that less that an eighth of the data recorded is actually necessary to create the visual effect we experience when looking at the picture. Beyond that, the quality is not horrible, either, but we can begin to detect bands forming due to the fact that the pixels were processed as distinct 8×8 blocks. By the end of the experiment, where the cutoff is 100, the picture is not good, yet there is still enough data that it is recognizable as a flower and shows a resemblance to the original image.

Solution to Problem 2: Compression is Fun and Easy

Solution to Problem 2, part a.

Table 1 represents the LZW analysis of the phrase "peter piper picked a peck of pickled peppers." The resulting data stream is:

```
80 70 65 74 65 72 20 70 69 82 86 88 63 6B 65 64 20 61 87 65 8D 20 6F 66 87 8C 6B 6C 8F 93 70
8A 73 81
```

This is 34 bytes long, whereas the original message was 46 bytes long (including the start and stop control characters), which amounts to 26.1% compression. If we count bits, the original message could have been sent in 7 bits per character (total 322 bits) whereas the LZW code requires 8 bits per character (total 272 bits) so the compression is 15.5%. Of course, this is a short example but contrived to make the dictionary grow quickly. For a sizeable selection of average English text, LZW typically yields 50% compression.

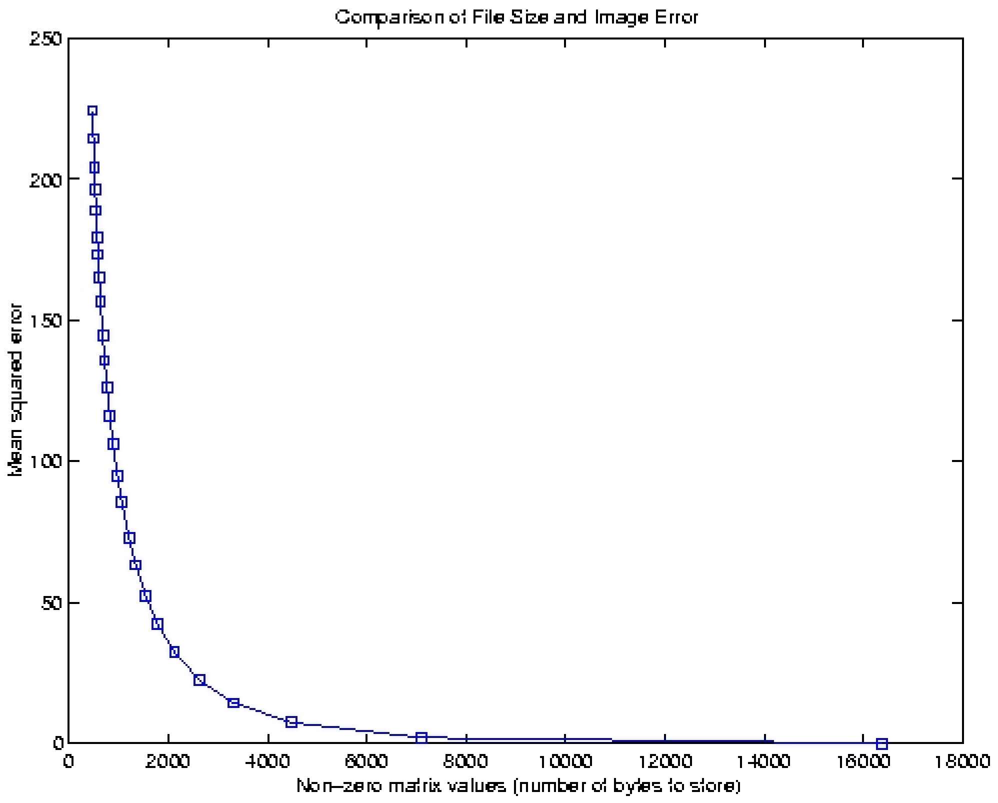

Figure 1: Comparison of File Size and Image Error

Solution to Problem 2, part b.

This strategy has several advantages: we will stay within 7-bit data, hence more compression and faster encoder/decoder; moreover, the dictionary can be searched quickly. There is a major drawback, though. The dictionary can only have 16 entries beyond the ASCII characters. Except for very short and repetitive sequences of characters, the dictionary will overflow. Another drawback is that the text cannot include any of the ASCII characters being used for the dictionary; among the characters excluded are three that are often used (DC1, DC3, and ESC). On the other hand, the characters HEX `80 - FF` would now be available since the dictionary no longer is there, and this includes many common accented letters of Western European languages. A practical implementation of the coder and decoder would not be either easier or harder by very much, but new programs would have to be written because this proposed arrangement is not standard, and new software always brings with it a cost and risk of bugs.

Input		New dictionary entry		Transmission		New dictionary entry		Output
02	STX	–	–	80	(start)	–	–	STX
70	p	–	–	–	–	–	–	–
65	e	82	pe	70	p	–	–	p
74	t	83	et	65	e	80	pe	e
65	e	84	te	74	t	81	et	t
72	r	85	er	65	e	82	te	e
20	space	86	r-space	72	r	83	er	r
70	p	87	space-p	20	space	84	r-space	space
69	i	88	pi	70	p	85	space-p	p
70	p	89	ip	69	i	86	pi	i
65	e	–	–	–	–	87	ip	–
72	r	8A	per	82	pe	–	–	pe
20	space	–	–	–	–	88	per	–
70	p	8B	r-space-p	86	r-space	–	–	r-space
69	i	–	–	–	–	89	r-space-p	–
63	c	8C	pic	88	pi	–	–	pi
6B	k	8D	ck	63	c	8A	pic	c
65	e	8E	ke	6B	k	8B	ck	k
64	d	8F	ed	65	e	8C	ke	e
20	space	90	d-space	64	d	8D	ed	d
61	a	91	space-a	20	space	8E	d-space	space
20	space	92	a-space	61	a	8F	space-a	a
70	p	–	–	–	–	90	a-space	–
65	e	93	space-pe	87	space-p	–	–	space-p
63	c	94	ec	65	e	91	space-pe	e
6B	k	–	–	–	–	92	ec	–
20	space	95	ck-space	8D	ck	–	–	ck
6F	o	96	space-o	20	space	93	ck-space	space
66	f	97	of	6F	o	94	space-o	o
20	space	98	f-space	66	f	95	of	f
70	p	–	–	–	–	96	f-space	–
69	i	99	space-pi	87	space-p	–	–	space-p
63	c	–	–	–	–	97	space-pi	–
6B	k	9A	ic	8C	pic	–	–	pic
6C	l	9B	ckl	6B	k	98	ic	k
65	e	9C	le	6C	l	99	ckl	l
64	d	–	–	–	–	9A	le	–
20	space	9D	ed-space	8F	ed	–	–	ed
70	p	–	–	–	–	9B	ed-space	–
65	e	–	–	–	–	–	–	–
70	p	9E	space-pep	93	space-pe	–	–	space-pe
70	p	9F	pp	70	p	9C	space-pep	pp
65	e	–	–	–	–	9D	pp	–
72	r	–	–	–	–	–	–	–
73	s	A0	pers	8A	per	–	–	per
03	ETX	–	–	73	s	9E	pers	s
–	–	–	–	81	(stop)	–	–	ETX

Table 1: Solution to Problem 2, part a