

## ECE 253a FINAL EXAM SOLUTION SET

### 1. Edge Sharpening (5 points)

**1:** `filter2(medium, test)`

This is (c), which is the only one that shows a blur (edges are rounded) with no sharpening.

**2:** `unsharp(test, small, 1)`

This is (d). We know that it is “small” because the overshoots and undershoots are narrow (as compared to (a) and (b) which are wider). And we know that it is “1” because the overshoots and undershoots are more accentuated (as compared to (e), where they are less strong).

**3:** `unsharp(test, small, 0.5)`

This is (e) for the same reasons as above.

**4:** `unsharp(test, large, 1)`

This is (a) for the same reasons as above.

**5:** `unsharp(filter2(medium, test), large, 1)`

This is (b), because it shows some of both the rounded-off edges from the blurring, and the sharpening effect.

The grading on this problem is 1 point per item.

### 2. Histograms and Averaging (5 points):

The input histograms have only two values. Let us take them to be 0 and 180, just for simplicity (since there will be some divisions by 9 coming up). When we filter the left hand image, the output histogram will now have 4 values: 0 and 180 as well as 60 and 120. The latter two occur when the 3x3 filter is centered on the dark side of the edge (6 dark pixels and 3 light ones), and centered on the light side of the edge (3 dark pixels and 6 light ones):

$$\frac{6 \times 0 + 3 \times 180}{9} = 60$$

$$\frac{3 \times 0 + 6 \times 180}{9} = 120$$

The exact heights of the bins can be calculated as:  $200 \times 99 = 19800$  pixels have value 0 (and the same for 180) and 200 pixels have value 60 (and the same for 120).

On the other hand, when we filter the checkerboard, we can have these 4 bin locations, as well as 2 others:

$$\frac{5 \times 0 + 4 \times 180}{9} = 80$$

$$\frac{4 \times 0 + 5 \times 180}{9} = 100$$

because the filter can be centered at one of the black/white corners, in which case there are 5 of one color and 4 of the other in the filter.

The exact heights of the bins can be calculated as follows: There are 64 squares total, so 32 dark squares. The squares are of size 25 by 25, so the interior portion of 23x23 will produce all dark output values.

$$32 \times 23 \times 23 = 16928$$

So there are 16928 dark pixels and the same for light pixels.

A pixel of value 60 occurs along the edges (but not the corners). There are 8 square edges vertically x 8 columns x 23 pixels in each x 2 because of horizontal edges. So there are 2944 pixels of value 60 (and the same for 120).

Lastly there are 64 corners, and each one has two pixels that will have a filtered output value of 80. So 128 pixels have a value 80, and 128 have a value 100.

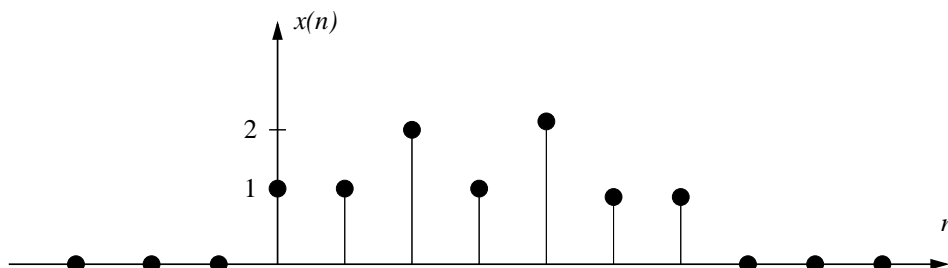
As a check,  $(16928 + 2944 + 128) \times 2 = 40,000$ .

NOTE: The heights of the bins that I gave here make the assumption that both images are padded outwards with the same type of pattern that they have on the inside. In particular, this means that the right-hand image continues with alternating blocks, and so the first and last row/column of blocks are no different from the interior blocks. If you make a different assumption about the image boundary, then you will get a different final answer, but we counted all such boundary assumptions as being correct.

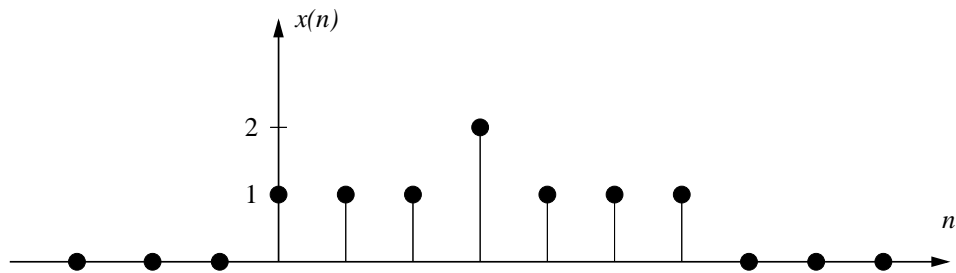
The grading for this problem is: You get 1 point for part (a) for simply recognizing that the histograms will not be the same. In part (b), you get 2 points for each histogram. This consists of 1 point for saying the number of bins, 1/2 point for saying the height of bins and 1/2 point for the location of bins.

### 3. Median Filtering (6 points):

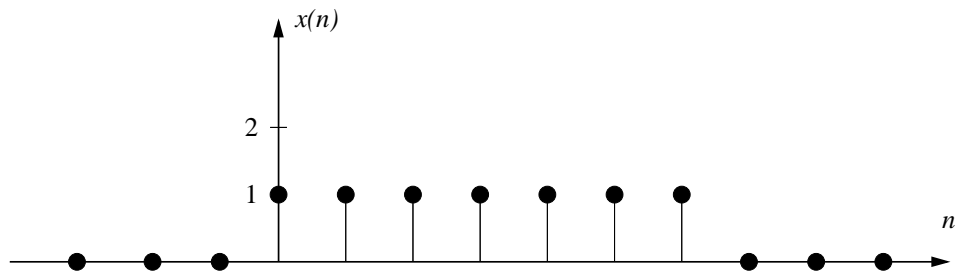
(a) After one iteration of median filtering, we have:



and after the 2nd we have:



and after the 3rd we have:



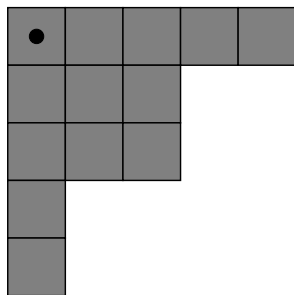
which is the root signal, because any subsequent median filtering (with this particular median filter) will cause no further change.

(b) If there are  $M^2$  noise pixels, then there must be at least  $M^2 + 1$  good pixels, so that the noise pixels will not constitute the majority in the median operation. So the entire mask must cover at least  $2M^2 + 1$  pixels. Thus the mask size must be at least  $N \times N$  where  $N$  is the next integer larger than  $\text{sqrt}(2M^2 + 1)$ .

The grading for this problem is 3 points for each part.

#### 4. Binary Morphology (5 points):

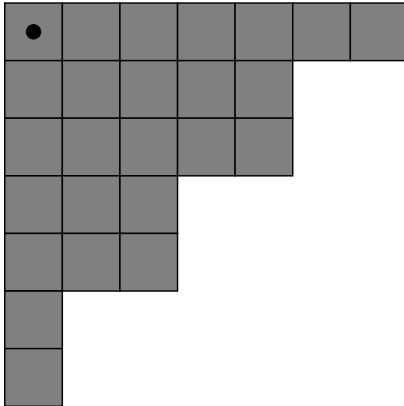
(a) The dilation of B by B is the set Y which looks like this:



(b) The smallest set  $X_1$  which could survive a *single* erosion by B is in fact the set B itself. That would leave just a single point behind, located at the origin of B.

The smallest set  $X_2$  which could survive *two* erosions by B is exactly the set Y which we found in part (a).

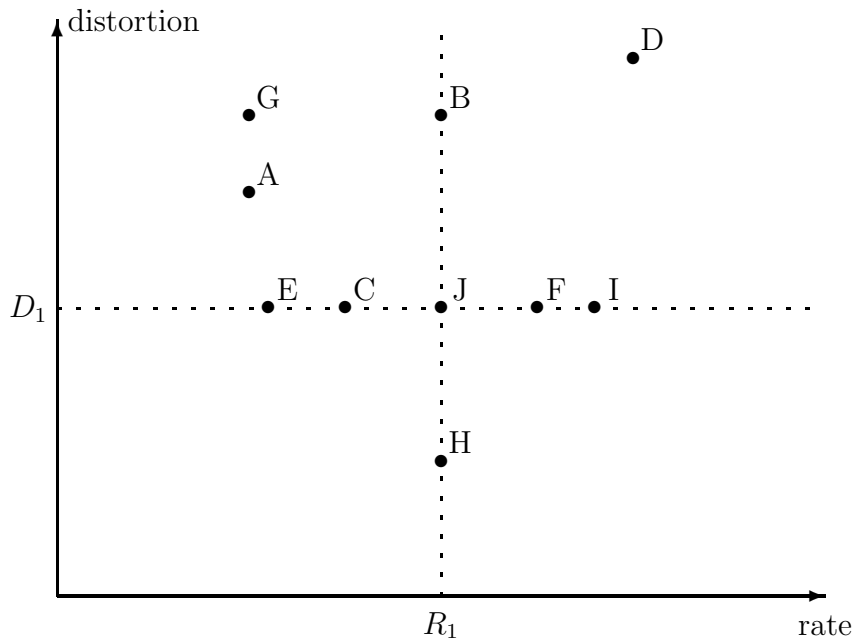
But we are looking for the smallest set which could survive *three* erosions. By dilating Y by B, we get a set which looks like this:



and this is the smallest set which can survive 3 erosions by B.

The grading for this is 2 points for part (a), and 3 for part (b). You can get partial credit for part (b) if you recognize, for example, that the set B is the smallest set which can survive one erosion by B, or that the set Y from part (a) can survive 2 erosions.

5. JPEG (9 points):



(A) Smoothing will make it more compressible, because the high frequencies are getting reduced in amplitude. So the rate will go down. The distortion however will go up.

(B) Because we are filtering afterwards, at the decoder, the compressed bit rate is unaffected. The distortion will be higher.

(C) In general, truncated Huffman coding incurs a penalty of slightly higher rate, compared to full Huffman coding. So we expect the rate will be slightly lower. The distortion however

will be unchanged, because a change to the Huffman coding alone cannot produce any effect on the quality of the picture.

**(D)** Usually, histogram equalization makes an image harder to compress with JPEG, because it makes it higher contrast. So we expect the rate to be higher. The distortion will definitely be higher as well, because the distortion is computed relative to the original image, so even the histogram equalized image alone, without any compression effects, would likely still have a high distortion relative to the original image.

**(E)** With improved prediction, the rate should decrease slightly, but the distortion is unaffected.

**(F)** Because of breaking the run-lengths of zeros, spectral selection progressive mode costs more bits (by the time you get to the end of the progression) than baseline sequential JPEG. The distortion is unchanged.

**(G)** If the encoder sets to zero some coefficients, then the rate will go down, and the distortion will go up. The encoder only does this for unimportant blocks, so the distortion as measured perceptually may be unchanged. But from the point of view of mean-squared error, the distortion definitely goes up if the encoder decided to zero out some coefficients beyond what the Q table says.

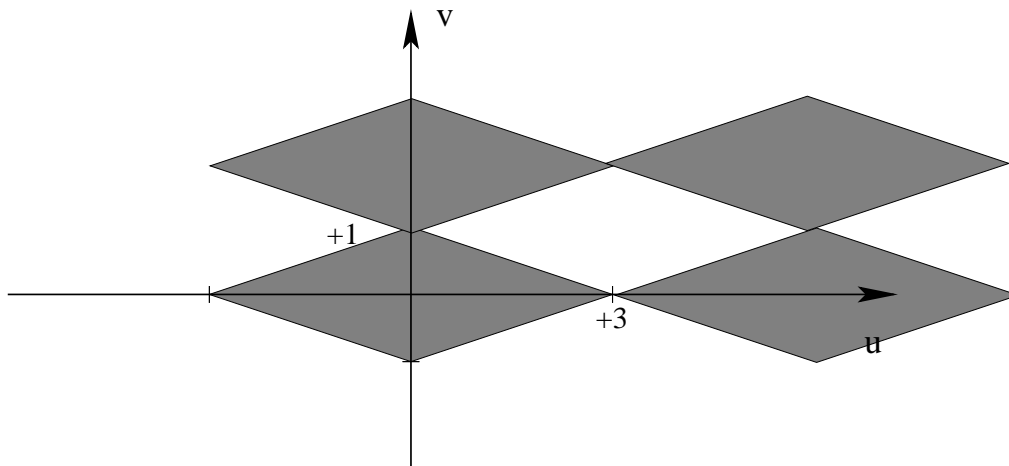
**(H)** The rate is unchanged, since this is a decoder-only operation. The distortion should go down a little.

**(I)** By multiplying all the step indices by 2, the rate will go up, because the Huffman codewords will tend to be longer. The distortion will be unchanged, because the decoder inverts the operation.

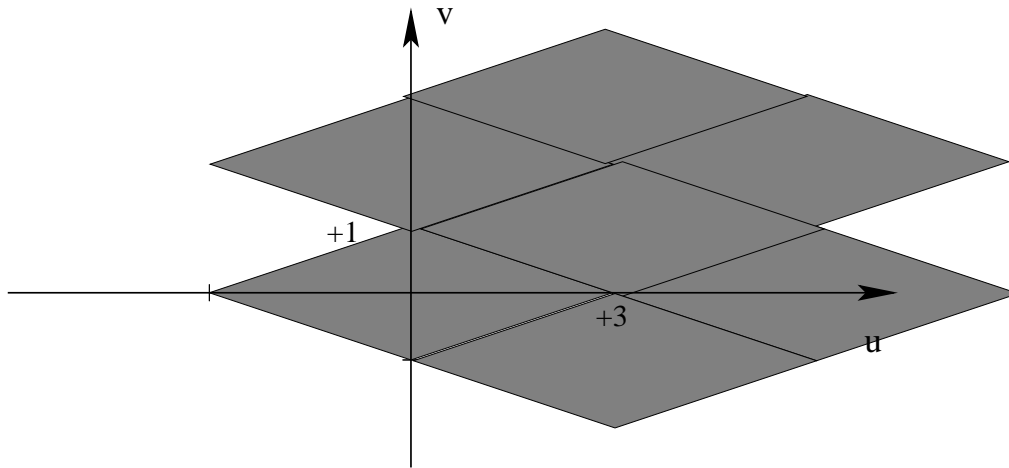
The grading on this problem is 1 point per item.

## 6. Sampling (6 points):

When we use a rectangular sampling grid, we will have a rectangular grid for spectral replications. The densest such one we can have, that will still ensure no aliasing, is this:



When we use a non-rectangular sampling, we will have a non-rectangular spectral replication lattice, and the densest such one we can have, that will still ensure no aliasing, is this:



In this problem, it is not necessary to do any calculations. One can see that the spectral replication lattice for the non-rectangular case is exactly twice as dense as the one for the rectangular case, because it has all the lattice points for the rectangular case, plus a set of “in-between” points. Therefore we can immediately conclude that the sampling lattice, in the spatial domain, will be half as dense. So, there is a 50% savings in sampling points. Full credit can be had for getting this answer in this manner, just using the picture.

Alternatively, one can carry out the calculations, in exactly the same manner as on page 8 of handout 11:

The generator matrix for the rectangular spectral replication lattice is:

$$B_{rect} = \begin{pmatrix} 6 & 0 \\ 0 & 2 \end{pmatrix}$$

and so the generator matrix for the sampling lattice in the spatial domain is

$$A_{rect} = (B_{rect}^{-1})^T = \begin{pmatrix} 1/6 & 0 \\ 0 & 1/2 \end{pmatrix}$$

and we have  $\det A_{rect} = \frac{1}{12}$

The generator matrix for the non-rectangular spectral replication lattice is:

$$B_{non-rect} = \begin{pmatrix} 3 & 0 \\ 1 & 2 \end{pmatrix} \text{ or } \begin{pmatrix} 6 & 3 \\ 0 & 1 \end{pmatrix}$$

Using the first of these, the generator matrix for the sampling lattice in the spatial domain is

$$A_{non-rect} = (B_{non-rect}^{-1})^T = \begin{pmatrix} \frac{1}{3} & \frac{-1}{6} \\ 0 & \frac{1}{2} \end{pmatrix}$$

and we have  $\det A_{non-rect} = \frac{1}{6}$

This lattice is less dense, and how much less dense can be determined by the ratio:

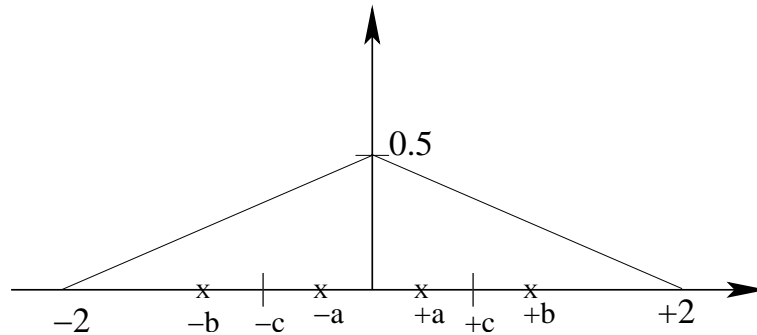
$$\frac{\text{sampling density of non-rect grid}}{\text{sampling density of rect grid}} = \frac{\frac{1}{\det A_{non-rect}}}{\frac{1}{\det A_{rect}}} = \frac{6}{12} = \frac{1}{2}$$

So the non-rectangular sampling grid is more efficient than rectangular sampling; it reduces the number of samples by 50%.

If you don't see the answer just from the picture, you can of course get full credit if you go through the steps above. You can get partial credit for having the correct replication lattices (2 points), stating the relationship between sampling and replication lattices (2 points) and having the relationship between the sampling density and the determinant of the generator matrix (2 points).

### 7. Quantization (6 points):

There are 4 reproduction levels, which be located at  $-b$ ,  $-a$ ,  $+a$ , and  $+b$ . There are 3 transition levels, which are located at  $-c$ ,  $0$ , and  $+c$ . The picture looks something like this:



For an optimal MSE quantizer, the transition levels must be halfway between the reproduction levels.  $0$  is already halfway between  $-a$  and  $+a$ . So the only additional requirement is that  $c = (a + b)/2$ .

For an optimal MSE quantizer, the reproduction levels must be the centroids of their regions. From this we get the conditions:

$$a = \frac{\int_0^c x f(x) dx}{\int_0^c f(x) dx}$$

$$b = \frac{\int_c^2 x f(x) dx}{\int_c^2 f(x) dx}$$

In these expressions,  $f(x)$  is the pdf function, which is (for the right-hand side)

$$f(x) = \frac{-1}{4}x + \frac{1}{2}$$

The grading for this problem is: 2 points for having the qualitatively correct picture of quantization levels and transition levels. One point each for the following 4 things:  $c = (a + b)/2$ , correct expression for a, correct expression for b, correct equation for the pdf.

### 8. Lempel-Ziv Coding (9 points):

(a) Consider first the sequence with period 3. We start off with the search buffer containing: abcabcab

The symbols which follow are: cabcabababcab

If we go back in the search window either 3 positions, or 6 positions, we will find a match that is *very* long. In fact, our match length is a number between 0 and 15, so unfortunately for this coder, the longest match is limited to 15.

The triple we encode will be of the form: <3, 15, next symbol>

This will take 12 bits to represent (3 bits for the offset, 4 for the match length, and 5 for the next symbol).

By encoding this triple, we convey 16 input symbols to the decoder. 15 of them through matching, and one through “next symbol.”

If we were to encode these 16 symbols using 5 bits each, it would require  $16 \times 5 = 80$  bits.

So the compression ratio is  $80/12$ .

Consider now the sequence of length 4. The search window contains abcdabcd and the lookahead window begins with abcdabcd etc. The situation is exactly the same as before. The compression ratio is  $80/12$ .

Things are different, however, with period 9. The search window contains abcdefgh and the lookahead window begins with iabcdefghiabcdefgh

The encoder cannot find i in the search window, so must encode this with the triple <1,0,i>. That is, the offset (1) is arbitrary, and the length of the match is 0, and the next symbol is i. The encoder uses 12 bits therefore to convey only 1 symbol! Without using the LZ coder, one uses only 5 bits per symbol. The compression ratio is  $5/12$  which means we have expansion, not compression. This will be true for any periodic sequence with period greater than 8.

(b) For the LZ78 style of encoder, you do not need to meticulously populate the entire dictionary. You can get an approximate value very quickly by reasoning as follows. The dictionary starts off with the entries a, b, and c. We then add all strings of length 2 to the dictionary (ab, bc, and ca) and all strings of length 3 (abc, bca, cab) and so forth. Since there are 256 entries in the dictionary, and  $256/3$  rounds down to 85, the dictionary at the end will have all strings of length 85. (Again, keep in mind that there are only 3 possible strings of length 85, since a string can only start with a or b or c.) So, when we encode with this fully populated

dictionary, 8 bits will convey 85 symbols. The compression ratio is  $(85 \times 5) / 8$ . This is VERY much better than part (a).

For the sequences with period 4, there are 4 strings of length 1, 4 of length 2, etc.  $256/4 = 64$ , so the compression ratio is  $(64 \times 5) / 8$ .

For the sequences with period 9, there are 9 strings of length 1, 9 of length 2, etc.  $256/9$  rounds down to 28, so the compression ratio will be at least  $(28 \times 5) / 8$ .

The LZ77 coder was causing expansion for a sequence of period 9 because the period was larger than the search buffer size. The LZ78 coder still gets substantial compression for these sequences.

This result is NOT meant to be taken as a general statement about the superiority of one type of coding. In fact, the comparison in this problem is not fair, because the LZ77 coder in part (a) was given a tiny search buffer (only size 8) whereas the LZ78 coder in part (b) was given a moderate size dictionary (256 entries). Furthermore, the LZ77 coder was hampered by the inclusion of “next symbol” in the triple, which, as discussed in class, is not the only way of doing this, and is not a particularly good way of coding things.

The grading for this problem is: 5 points for part (a), 4 points for part (b).

## 9. Color (6 points):

(a) Here, we should picture the RGB cube (normalized to have sides of length 1), and the pixel A (where the RGB values have been divided by 255, so they have values between 0 and 1) has a color which is some vector inside the cube. The hue and saturation are determined entirely by where this vector intersects the plane  $R+G+B = 1$  (or, if the vector is too short to intersect the plane, then we have to extend the vector until it does intersect).

Multiplying by 2 will not change the point of intersection. So, the hue and saturation do not change.

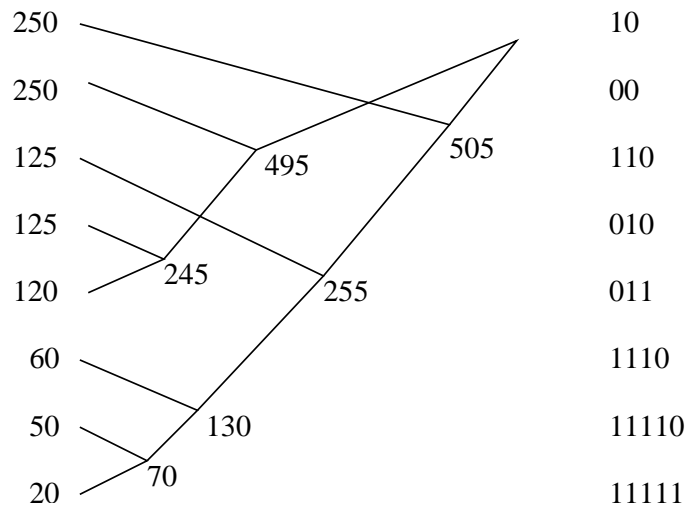
(b) Even though the histogram equalization transformation functions are identical, we did not say that the RGB values of pixel A are identical. Suppose they are  $R=1, G=50, B=99$ . And suppose that the histogram equalization transformation remaps 1 to be 1, 50 to be 100, and 99 to be 255. The vector in this case will not intersect the plane at the same point. Both the hue and saturation can be changed by this operation.

(c) The reference white for the NTSC RGB color coordinate system is given by  $(R_n, G_n, B_n) = (1,1,1)$ . Likewise, the reference white for the CIE RGB color coordinate system is given by  $(R_c, G_c, B_c) = (1,1,1)$ . The last page of your handout on color (handout 6) had the  $3 \times 3$  conversion matrices for going from one color coordinate system to another. By looking at just one row of the  $3 \times 3$  conversion matrix, it can easily be verified that the vector  $(1,1,1)$  in the  $(R_n, G_n, B_n)$  system does not convert to the vector  $(1,1,1)$  in the  $(R_c, G_c, B_c)$  system. So they are not the same color.

The grading for this problem is: 2 points for each part.

10. **Huffman Coding (9 points):**

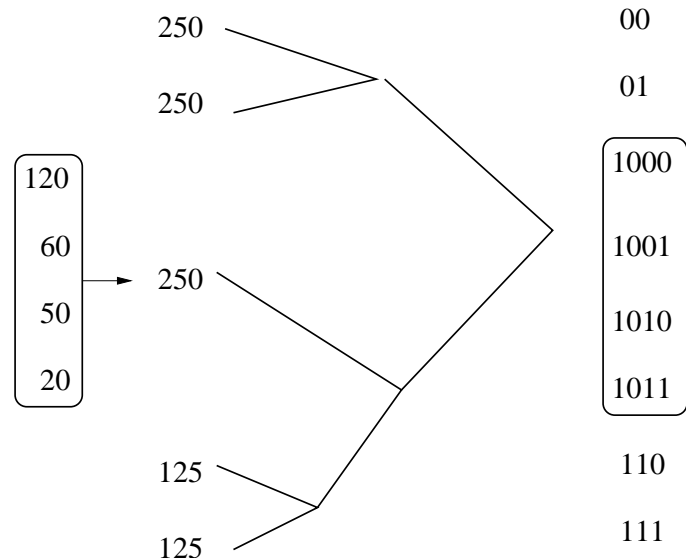
(a) The Huffman design procedure and the resulting codewords look like this:



The expected length of the code is

$$E(l) = 0.25 \times 2 + 0.25 \times 2 + 0.125 \times 3 + 0.125 \times 3 + 0.12 \times 3 + 0.06 \times 4 + 0.05 \times 5 + 0.02 \times 5 = 2.7 \text{ bits/pixel}$$

(b) For the truncated code, we group together the least probable symbols, which together have a count of 250 pixels, or a probability of 0.25. The Huffman design procedure is shown below. To get the actual codewords, we need to append 2 bits to those symbols which are part of the group. So the final codewords are as listed below.



The expected length of this code is:

$$E(l) = 0.25 \times 2 + 0.25 \times 2 + 0.125 \times 3 + 0.125 \times 3 + 0.25 \times 4 = 2.75$$

