

## Homework Assignment II

1. (50 points) Consider a source  $\mathcal{A}$  that generates symbols from an alphabet of size four,  $\mathcal{A} = \{a, b, c, d\}$ , with probabilities

$$P(a) = \frac{1}{2}, \quad P(b) = \frac{1}{4}, \quad P(c) = \frac{1}{8}, \quad \text{and} \quad P(d) = \frac{1}{8}.$$

- a. Find the binary arithmetic code for the sequence *aabcbad*. Show all of your encoding steps along with the coding intervals.
  - b. Decode the binary code above and verify that your encoding is correct.
  - c. Repeat the encoding/decoding process for the adaptive case where we do not have any symbol statistic in the beginning. Compare the efficiency of the two cases.
2. **Computer Assignment** (50 points). In this assignment, you are asked to implement a Huffman coding/decoding algorithm. Feel free to use the codes in our in-class demonstrations (with your own modification of course).
    - a. Write a subroutine (`huff.c` or `huff.m`) to construct Huffman codes. The input is the probability distribution of the symbols. The output is an ASCII Huffman table with the symbols and their corresponding Huffman codes.
    - b. Write a program to encode a file using Huffman coding. Treat the input file as a sequence of bytes, each byte as an `unsigned char`. Hence, the set of symbols is  $\{0, 1, 2, \dots, 255\}$ . This program can be a two-pass process. In the first pass, frequency of each symbol is collected. In the second pass, string of symbols are converted into cascaded codewords which are saved into an output file. If the total number of bits is not a multiple of 8, pad enough zero bits at the end. You should also save the Huffman table into a separate file.
    - c. Write the corresponding Huffman decoder. The inputs are the compressed bitstream and the Huffman table. The command lines should look like this:

```
huffmanencode filename huffman_table compressed_bitstream
huffmandecode compressed_bitstream huffman_table filename_rec
```
    - d. Test your Huffman codec on the first 10 frames of the `glasgow` sequence. Is the compression lossless? How much compression can you achieve on the first 10 frames, not counting the Huffman tables?

e. Now, instead of coding each frame independently, Huffman encode the difference between sequential frames. Say, encode Frame 1, then Frame 2 - Frame 1, Frame 3 - Frame 2, Frame 4 - Frame 3, etc. Write the appropriate decoder for this algorithm. Can you achieve losslessness? Compare your result with Part d's above. Which scheme offers better compression?

Due date: **Feb 21** in class