1. Consider the simple Bernoulli model that generates independent random bits with \( \Pr[X_i = 1] = p \) for some fixed \( 0 \leq p \leq 1 \). For sequence length \( n \), and some \( \epsilon > 0 \), the typical set \( A_n^\epsilon \) is defined as the set of sequences \( x_1, \ldots, x_n \) such that

\[
2^{-n(H(X)+\epsilon)} \leq p(x_1, \ldots, x_n) \leq 2^{-n(H(X)-\epsilon)}.
\]

What are the sequences in the typical set \( A_{15}^{0.1} \) under the Bernoulli model when \( p = 0.1 \)? How about \( p = 0.3 \), and \( p = 0.5 \)? What can you say about the sizes of these sets?

2. Given a set of (source) symbols, \( x_1, \ldots, x_m \) and the corresponding probabilities, \( p_1, \ldots, p_m \), so that \( \Pr[X = x_i] = p_i \), the Shannon-Fano code works as follows:

1. Sort the symbols according to decreasing probability so that we can assume \( p_1 \geq p_2 \geq \ldots \geq p_m \).
2. Initialize all codewords \( w_1, \ldots, w_m \) as the empty string.
3. Split the symbols in two sets, \( (x_1, \ldots, x_k) \) and \( (x_{k+1}, \ldots, x_m) \), so that the total probabilities of the two sets are as equal as possible, i.e., minimize the difference \( |(p_1 + \ldots + p_k) - (p_{k+1} + \ldots + p_m)| \).
4. Add the bit ‘0’ to all codewords in the first set, \( w_i \mapsto w_i0 \), for all \( 1 \leq i \leq k \), and ‘1’ to all codewords in the second set, \( w_i \mapsto w_i1 \) for all \( k < i \leq m \).
5. Keep splitting both sets recursively (Step 3) until each set contains only a single symbol.

Simulate the Shannon-Fano code, either on paper or in silico, for a source with symbols \( A : 0.2, B : 0.22, C : 0.25, D : 0.15, E : 0.13, F : 0.05 \), where the numbers indicate the probabilities \( p_i \).

3. Take a piece of text, estimate the symbol occurrence probabilities from it. Then use them to encode the text using the Shannon-Fano code (on a computer). Compare the code-length to the entropy of the symbol distribution \( H(X) \).

4. (About Friday’s guest lecture:) What is the complexity vs. goodness-of-fit trade-off? In other words, why is it a bad idea to encode data using a) a very simple model, or b) a very complex model? Do you think “complexity” (of a model) is a well-defined concept, and why (not)?

Bonus exercise. A variation of Exercise 3: What if you consider each pair of characters in the text as one symbol, and encode them using one codeword? What happens to the total code-length? Can you do it for triplets (three symbols at a time)?