Please submit the following problem solutions for grading:

1. Cover and Thomas: 5.13

The average length of the “code” employed by player $B$ has average length $L^* = 38.5$. We know that the optimal code satisfies

$$L^* \leq H(X) + 1 \leq \log |X| + 1.$$ 

This implies

$$|X| \geq 2^{L^*-1} = 2^{37.5} = 1.9 \times 10^{11}.$$ 

2. Cover and Thomas: 5.17

(a) We need to choose the codeword lengths $l_i$ to minimize the $C = \sum_i p_i c_i l_i$ subject to the constraint $\sum_i 2^{-l_i} \leq 1$. We can solve this problem by casting it as a problem whose solution is already known. Let $B = \sum_i p_i c_i$. Since $B$ is a constant, we can equivalently minimize

$$C' = C/B = \sum_i \frac{p_i c_i}{B} l_i = \sum_i p'_i l_i.$$ 

Note that $p'_i = p_i c_i/B$ is a probability distribution. Thus when we ignore the integer constraints on $l_i$, minimizing $C'$ is precisely the problem solved at the start of Section 5.3. The solution is

$$l^*_i = -\log p'_i = -\log \frac{p_i c_i}{\sum_j p_j c_j}.$$ 

The minimum cost associated with these non-integer codeword lengths is

$$C^* = \sum_i p_i c_i (-\log p'_i) = BH(p'_i).$$ 

(b) All we need to do is use the probabilities $p'_i$ for the Huffman procedure.

(c) We can upperbound the Huffman procedure by upper bounding the lengths of the optimal codewords. In particular, we know from the Kraft inequality that the Huffman code will have codewords with lengths $l_i$ satisfying

$$-\log p'_i \leq l_i \leq \lceil -\log p'_i \rceil \leq -\log p'_i + 1.$$ 

Multiplying by $p'_i$ and summing over all $i$, we obtain

$$-\sum_i p'_i \log p'_i \leq \sum_i p'_i l_i \leq -\sum_i p'_i \log p'_i + 1.$$ 

Since $p'_i = p_i c_i/B$, we multiply through by $B$ to obtain

$$C^* \leq \sum_i p_i c_i l_i \leq C^* + B.$$ 

1
3. RLL Coding

(a) \( Y_n \) denote the number of source symbols used to map to the \( n \)th intermediate digit. Since there are \( N(t) \) intermediate digits produced by time \( t \),
\[
\frac{Y_1 + \cdots + Y_{N(t)}}{N(t)} \leq \frac{t}{N(t)} \leq \frac{Y_1 + \cdots + Y_{N(t)+1}}{N(t)} = \frac{N(t) + 1 Y_1 + \cdots + Y_{N(t)+1}}{N(t) + 1}.
\]
Since the \( X_i \) are iid, the \( Y_i \) are iid. By the law of large numbers,
\[
\lim_{n \to \infty} \frac{Y_1 + \cdots + Y_n}{n} = E[Y].
\]
Since \( N(t) \to \infty \) as \( t \to \infty \), it follows that
\[
\lim_{t \to \infty} \frac{N(t)}{N(t) + 1} = 1
\]
and
\[
\lim_{t \to \infty} \frac{Y_1 + \cdots + Y_{N(t)}}{N(t)} = E[Y].
\]
It follows that
\[
E[Y] \leq \lim_{t \to \infty} \frac{t}{N(t)} \leq E[Y].
\]
Note that \( Y \) has a truncated geometric PMF. For this problem, its easier to work with the complementary CDF of \( Y \). In particular,
\[
P[Y > j] = \begin{cases} 1 & j < 0 \\ q^j & j = 0, 1, \ldots, 7 \\ 0 & \text{otherwise} \end{cases}
\]
The expected value of \( Y \) is
\[
E[Y] = \sum_{j=0}^{\infty} P[Y > j] = \sum_{j=0}^{7} q^j = \frac{1 - q^8}{1 - q}
\]
Finally, if \( t/N(t) \) is converging to a constant, then \( N(t)/t \) must converge to its reciprocal. Thus
\[
\alpha = \lim_{t \to \infty} \frac{N(t)}{t} = \frac{1}{E[Y]} = \frac{1 - q}{1 - q^8}.
\]

(b) Let \( R_k \) denote the number of code bits produced by the \( k \)th intermediate digit. Note that \( R = 1 \) if intermediate digit 8 is produced; otherwise, \( R = 4 \). Since the probability that intermediate digit 8 is produced is \( q^8 \),
\[
\beta = E[R] = 4(1 - q^8) + q^8 = 4 - 3q^8
\]
Note that
$$\hat{M}(k) = R_1 + \cdots + R_k$$

Since $R_1, R_2, \ldots$ is an iid random sequence, the strong law of large number says that
$$\beta = \lim_{k \to \infty} \frac{\hat{M}(k)}{k} = E[R] = 4 - 3q^8 \quad \text{w.p. 1}$$

(c) Note that
$$\frac{M(t)}{t} = \frac{\hat{M}(N(t))}{N(t)} N(t).$$

Since $N(t) \to \infty$ as $t \to \infty$, it follows that
$$\lim_{t \to \infty} \frac{M(t)}{t} = \lim_{t \to \infty} \frac{\hat{M}(N(t))}{N(t)} = \frac{\alpha \beta}{1 - q^8} = \frac{(1 - q)(4 - 3q^8)}{1 - q^8}.$$

(d) For the RLL code, the average number of code symbols per unit time is
$$L_{RLL}^* = \alpha \beta \frac{(1 - q)(4 - 3q^8)}{1 - q^8}.$$  

Comparing this performance to a Huffman code that encodes four source digits at a time can be tricky if we don’t know the value of $q$. Since the Huffman code is optimal and since we are coding 4 source symbols per codeword, we know that the average number of code bits per source bit satisfies
$$H(q) \leq L_{RLL}^* \leq H(q) + 1/4.$$  

However, we will see shortly that this bound is somewhat ambiguous. Ideally, we would like to find the Huffman code for each value of $q$ and compare $L_{H}^*$ against $L_{RLL}^*$. However this is either tricky or a LOT of work since the Huffman tree needs to be computed for each $q$. However, the RLL technique only makes sense if $q$ is high. In this case, we can make some assumptions about $q$. Specifically, let $S_k$ denote the set of 4 bit source sequences $x$ with $k$ zeros. We assume that $q$ is sufficiently large to guarantee that for any sequence $x \in S_k$,
$$P[x] \geq \sum_{k' < k, x' \in S_{k'}} P[x'].$$

That is, we assume each source sequence with $k$ zeros has higher probability than the combined probability of all sequences with fewer than $k$ zeros. This yields the following set of requirements on $q$:

\begin{align*}
  k = 1 & \quad q(1 - q)^3 > (1 - q)^4 \quad \Rightarrow q > 1/2 \\
  k = 2 & \quad q^2(1 - q)^2 > 4q(1-q)^3 + (1-q)^4 \quad \Rightarrow q > \frac{1 + \sqrt{5}}{4} = 0.809 \\
  k = 3 & \quad q^3(1-q) > 6q^2(1-q)^2 + 4q(1-q)^3 + (1-q)^4 \quad \Rightarrow q \geq 0.869 \\
  k = 4 & \quad q^4 > 1 - q^4 \quad \Rightarrow q > (1/2)^{1/4} = 0.8409
\end{align*}
Thus, for $q \geq 0.869$, the Huffman code tree has the property that all nodes with $k$ zeros are tied together before any node with $k + 1$ zeros is tied in. In this case, an example of the resulting tree is

Using $L(x)$ to denote the length of the codeword assigned to source sequence $x$, the average codeword length is

$$L^*_H = \sum_{k=0}^{4} \sum_{x \in S_k} L(x) P[x]$$

$$= \sum_{k=0}^{4} q^{4-k}(1-q)^k \sum_{x \in S_k} L(x)$$

$$= q^4 + 13q^3(1-q) + 41q^2(1-q)^2 + 37q(1-q)^3 + 10(1-q)^4$$

The average number of code bits per source bit is $L^*/4$. The following plots compares the Huffman code and the RLL code as a function of $q$. 

4
The RLL code is better for large $q$. This emphasizes that the Huffman code is optimal only for mapping a fixed set of symbols (4 source symbols in this case) to variable length codewords.