1. (3 points)
Marking:
2 marks for part (a.), 1 mark for part (b.).

(a) What is the "inverted file" of the following: \texttt{abccbaabc}?

\textbf{Solution:}

\begin{center}
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline
symbol & $n_i$ & $t_i$ \\
\hline
a & 3 & 1 & 0 & 0 & 0 & 0 \\
b & 3 & 0 & 1 & 0 & 0 & 1 \\
c & 3 & 0 & 0 & 1 & 1 & 0 \\
\hline
\end{tabular}
\end{center}

We can also encode the gaps between instances of a symbol.

\begin{center}
\begin{tabular}{|c|c|c|}
\hline
symbol & $n_i$ & offset \\
\hline
a & 3 & 1 \\
b & 3 & 2 \\
c & 3 & 3 \\
\hline
\end{tabular}
\end{center}

(b) How can "header" information be used to compress inverted files?

\textbf{Solution:}

We use gap or run-length encoding. Define a prefix code for the gaps of each list (say use a Golomb code). The header file needs to send $n$ and $n_i$ for each list. CHECK THIS IN NOTES.
2. (4 points)

(a) Encode the following sequence using LZ2 (sliding window):

baabaaaabb

Assume the window size is $n_w = 4$. Your answer should include a parsing of the above sequence into phrases.

Solution:
The phrase is parsed as follows: $b, a, a, baa, aa, b, b$.

<table>
<thead>
<tr>
<th>length</th>
<th>symbol</th>
<th>offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>'b'</td>
<td>-</td>
</tr>
<tr>
<td>0</td>
<td>'a'</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>'b'</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>1</td>
</tr>
</tbody>
</table>

Let $C_0$ be the code for phrase length, $C_1$ be the code for offset, and $C_2$ be the code for symbols. This yields the following code:

$C_0(0)C_2(b)C_0(0)C_2(a)C_0(1)C_1(1)C_0(3)C_1(3)C_0(2)C_1(2)C_0(0)C_2(b)C_0(1)C_1(1)$

(b) For the same sequence as in (a), show the phrase table built by an LZ3 encoder.

Solution:

<table>
<thead>
<tr>
<th>position</th>
<th>symbol</th>
<th>parent</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>'b'</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>'a'</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>'b'</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>'a'</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>'b'</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>'b'</td>
<td>0</td>
</tr>
</tbody>
</table>

3. (3 points)

(a) If the alphabet is $\{0, 1\}$, then a Lempel Ziv algorithm takes any finite sequence of bits (a sequence) and maps it to another finite sequence of bits (a codeword for the sequence). Thus, the LZ algorithm defines a code on the set of finite bit strings. This code is not a prefix code. Why not? Give a counter example (and
specify the LZ method).

**Solution:**
In general the Lempel Ziv algorithm does not define a prefix code because if we have two strings \( s_1 \) and \( s_2 \), such that \( s_1 \) is a prefix of \( s_2 \), then the code for \( s_1 \), \( \text{LZ}(s_1) \), is a prefix of the code for \( s_2 \), \( \text{LZ}(s_2) \).

For a specific counter example let’s use LZII to encode strings \( s_1 = 0 \) and \( s_2 = 00 \). Let \( C_0 \) be the code for phrase length, \( C_1 \) be the code for offset, and \( C_2 \) be the code for symbols.

\[
\text{LZII}(s_1) = C_0(0)C_2(0) \\
\text{LZII}(s_2) = C_0(0)C_2(0)C_0(1)C_1(0)
\]

Notice that the code for \( s_1 \) is a prefix for the code for \( s_2 \).

(b) How could one modify the LZ algorithm/code so that it does define a prefix code?

**Solution:**
There are several ways to do this:

- Have the algorithm encode the length of the sequence and send it at the end (or beginning) of the encoded sequence.
- Introduce a 'null' character that is encoded at the end of the encoded sequence. Note that the codeword for this 'null' character must not be a prefix of any codeword in \( C_0 \).

4. **(2 points)**

Consider the following conditional probability matrix for a first order Markov model, where the number of symbols in the alphabet is \( N = 3 \). Assume that these conditional probabilities are the same for all \( j \).

\[
P(X_{j+1} \mid X_j) = \frac{1}{8} \begin{bmatrix} 1 & 3 & 2 \\ 1 & 4 & 1 \\ 6 & 1 & 5 \end{bmatrix}
\]

If the marginal probability \( p(X_1) \) is uniform, then what are the marginal probabilities \( p(X_2) \) and \( p(X_3) \) ?

**Solution:**
For this we use the formula for marginal probability:

\[ p(X_{j+1}) = p(X_{j+1} | X_j)p(X_j) \]

\[
\begin{align*}
p(X_2) &= p(X_2 | X_1)p(X_1) = \frac{1}{8} \begin{bmatrix} 1 & 3 & 2 \\ 1 & 4 & 1 \\ 6 & 1 & 5 \end{bmatrix} \begin{bmatrix} \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{2} \end{bmatrix} = \begin{bmatrix} \frac{1}{4} \\ \frac{1}{4} \\ \frac{1}{2} \end{bmatrix} \\
p(X_3) &= p(X_3 | X_2)p(X_2) = \frac{1}{8} \begin{bmatrix} 1 & 3 & 2 \\ 1 & 4 & 1 \\ 6 & 1 & 5 \end{bmatrix} \begin{bmatrix} \frac{1}{3} \\ \frac{1}{4} \\ \frac{1}{2} \end{bmatrix} = \begin{bmatrix} \frac{7}{32} \\ \frac{17}{32} \end{bmatrix}
\end{align*}
\]

5. **(3 points)**

The following trie shows the frequency counts of a sequence, after \( j \) symbols have been seen by the encoder.

(a) Estimate the conditional probability of \( X_{j+1} \), assuming:

- a 0\textsuperscript{th} order Markov model;
- a 1\textsuperscript{st} order Markov model;

You may assume that the alphabet is \( \{a, b\} \) only. Be sure to state any other assumptions you make.

Hint: the count at a node at level \( k + 1 \) is the number of times that the symbol at that node is the next symbol, given the previous \( k \) symbols.
Solution:
The 0th order Markov model is:

\[
P(X_{j+1}) = \frac{1}{125} \begin{bmatrix} 41 \\ 84 \end{bmatrix}
\]

The 1st order Markov model is:

\[
P(X_{j+1} \mid X_j) = \begin{bmatrix} \frac{31}{41} & \frac{5}{84} \\ \frac{10}{41} & \frac{78}{84} \end{bmatrix}
\]

(b) What were the symbols \(X_j\) and \(X_{j-1}\)?

Solution:
This was a trick question. To find out \(X_j\) and \(X_{j-1}\) you simply needed to see where the counts did not add up. For level \(k = 1\) we can see that \(78 + 5 \neq 84\) \((X_j = b)\), and at level \(k = 2\) that \(60 + 17 \neq 78\) \((X_{j-1} = b)\).