Exercises 2 (solutions) COMP 423

Move to front

1. (a) LIKE I KNOW YOU LIKE COFFEE LIKE YOU KNOW I LIKE COFFEE

   (b) There are at least two possible ways of doing this.

   - Encode the word as ASCII string, with the ASCII NULL character for the end of word.
   - Encode the length of the word, and then encode the word as a sequence of ASCII characters (without an null termination).

2. On the left below are the codes for the words in the sequence. On the right is the list after each word has been encoded.

$C(1)$ C_{raw}(`THE`)	`THE`
$C(2)$ C_{raw}(`CAR`)	`CAR, THE`
$C(3)$ C_{raw}(`ON`)	`ON, CAR, THE`
$C(3)$	`THE, ON, CAR`
$C(4)$ C_{raw}(`LEFT`)	`LEFT, THE, ON, CAR`
$C(5)$ C_{raw}(`HIT`)	`HIT, LEFT, THE, ON, CAR`
$C(3)$	`THE, HIT, LEFT, ON, CAR`
$C(5)$	`CAR, THE, HIT, LEFT, ON`
$C(6)$ C_{raw}(`I`)	`I, CAR, THE, HIT, LEFT, ON`
$C(5)$	`LEFT, I, CAR, THE, HIT, ON`

   Make sure that you could decode the original sequence given the following sequence of codewords:

   $C(1)$ C_{raw}(`THE`) $C(2)$ C_{raw}(`CAR`) $C(3)$ C_{raw}(`ON`) $C(3)$ $C(4)$ C_{raw}(`LEFT`) $C(5)$ C_{raw}(`HIT`) $C(3)$ $C(5)$ $C(6)$ C_{raw}(`I`) $C(5)$

3. (a) When using move-to-front, we encode the position of the symbol within the list. Since the position cannot be 1 (the same symbol can never be repeated consecutively), we only need to encode positions 2 through N. This reduces the "alphabet" of positions and lets the codewords for positions be optimally chosen for this reduced alphabet.

   (b)
$$\sum_{i \neq k} p(X_{j+1} = A_i \mid X_j = A_k\ ) = (N-1)p_0 = 1$$

$$p_0 = \frac{1}{N-1}$$

   Since any of the N-1 symbols have equal probability, we use a fixed length code, using $\log\lceil N-1\rceil$ bits per symbol.

4. Worst-case upper bound on number of bits used for each Elias-encoded inverted list is as follows. For each symbol in an inverted file we encode the location of the first occurrence, and then the gap sizes between successive occurrences: $[n_i : t_1^i, (t_2^i - t_1^i)...]$

Take a convex function $f(x) = 2\log j + 1$ such that $\lambda(j) \leq f(j)$ where j is a positive integer.

Now we have,

$$\lambda^i_{tot} \leq f(t^i_1) + f(t^i_2 - t^i_1) + ... + f(t^i_{n_i} - t^i_{n_{i-1}})$$

Multiply by $\frac{n_i}{n_i}$ and apply Jensen's inequality to get:

$$\begin{aligned}
\lambda^i_{tot} &\leq n_i \frac{1}{n_i}(f(t^i_1) + f(t^i_2 - t^i_1) + ... + f(t^i_{n_i} - t^i_{n_{i-1}})) \\
&\leq n_i f(\frac{1}{n_i}(t^i_1 + (t^i_2 - t^i_1) + ... + (t^i_{n_i} - t^i_{n_{i-1}}))) \\
&= n_i f(\frac{t^i_{n_i}}{n_i}) \\
&\leq n_i f(\frac{n}{n_i}) \\
&= n_i(2\log(\frac{n}{n_i}) + 1)
\end{aligned}$$

For an upper bound on the average codeword length:

$$\begin{aligned}
\lambda_{avg} &= \frac{1}{n}\sum_{i=1}^{N} \lambda^i_{tot} \leq \sum_{i=1}^{N} \frac{n_i}{n} f(\frac{n}{n_i}) = \sum_{i=1}^{N} \frac{n_i}{n}(2\log(\frac{n}{n_i}) + 1) \\
\lambda_{avg} &= 2H + 1
\end{aligned}$$

Lempel Ziv

1. (a)

`R, A, L, F, #, A, T, E, #A, LF, ALFA`

Length	Offset	Symbol
0		R
0		A
0		L
0		F
0		#
1	4	
0		T
0		E
2	4	
2	8	
4	3	

(b)

`R, A, L, F, #, AT, E, #A, LF, AL, FA`

phrase	parent	newsymbol
1	0	R
2	0	A
3	0	L
4	0	F
5	0	$\#$
6	2	T
7	0	E
8	5	A
9	3	F
10	2	L
11	4	A

2. The Lempel-Ziv encoder could take two passes. In the first pass, it could count the frequencies of the match lengths or the offsets. Then it could make a Huffman code for either (or both) of these. It would need to send this Huffman code to the decoder as a header file.

3. (a) The decoding of the original sequence using LZ2 yields : `abaabbabab`

   (b) There are several correct sequences for this question, depending on the order of tree traversal. The necessary condition is that your sequence must be made up of 9 individual subsequences representing the 9 internal nodes/leaves (minus the root) of the tree, while traversing the tree from top to bottom. A possible sequence of symbols is: `a b aa ab ba bb abb bab bbb`

   (c) The decoding yields: `abacccaaca`

4.

   (a)

$Length$	$Offset$	$Symbol$
0	-	'0'
3	1	-
0	-	'1'
20	5	-

   (b) To encode a raw symbol we need $\lceil \log N \rceil$ bits, where N is the size of the alphabet. To encode an offset we need $\lceil \log n_w \rceil$ bits where n_w is the size of the offset. Therefore for the offset code to use fewer bits than the raw symbol code we need $m > \log n_w / \log N$, where m is the size of the match.

   NOTE: It doesn't matter how the length of the match should be encoded since both the offset and raw code require that length of match is coded. So they use the same number of bits for the length of match.

   (c)

Index	Index of parent/prefix	Last Symbol
0	null	
1	0	0
2	1	0
3	1	1
4	2	0
5	3	1
6	4	1
7	4	0
8	0	1
9	7	*

Markov models

1. (a) The conditional probability function $P(X_{i+1}|X_i)$ on symbols A_1, A_2, A_3 can be represented by the 3×3 matrix P

$$P = \begin{bmatrix} \frac{3}{4} & 18 & 18 \\ 18 & 34 & 18 \\ 18 & 18 & 34 \end{bmatrix}$$

such that $P_{jk} = P(X_{i+1} = A_j | X_i = A_k)$.
In general, the marginal probabilities $p(X_i)$ and $p(X_{i+1})$ need not be the same. But in the special case where we have stationary sequence of random variables, they have to satisfy

$$\mathbf{v} = P(X_{i+1}|\ X_i)\ \mathbf{v}$$

that is, they have to be eigenvectors of $P(X_{i+1}|\ X_i)$ with eigenvalue 1.
Computing the eigenvectors of the given $P(X_{i+1}|\ X_i)$ yields:

$$p(X_{i+1}) = p(X_i) = \begin{bmatrix} 13 \\ 13 \\ 13 \end{bmatrix}$$

The joint probability matrix is :

$$\begin{bmatrix} 34 & 18 & 18 \\ 18 & 34 & 18 \\ 18 & 18 & 34 \end{bmatrix} \begin{bmatrix} 13 & 0 & 0 \\ 0 & 13 & 0 \\ 0 & 0 & 13 \end{bmatrix} = \begin{bmatrix} 14 & 124 & 124 \\ 124 & 14 & 124 \\ 124 & 124 & 14 \end{bmatrix}$$

(b) Assuming a stationary sequence of symbols, then we must have.

$$p(X_2) = Pp(X_1) = p(X_1)$$

However :

$$p(X_2) = P\ p(X_1) = \begin{bmatrix} \frac{3}{4} & \frac{1}{8} & \frac{1}{8} \\ \frac{1}{8} & \frac{3}{4} & \frac{1}{8} \\ \frac{1}{8} & \frac{1}{8} & \frac{3}{4} \end{bmatrix} \begin{bmatrix} \frac{1}{4} \\ \frac{1}{4} \\ \frac{1}{2} \end{bmatrix} = \begin{bmatrix} \frac{9}{32} \\ \frac{9}{32} \\ \frac{14}{32} \end{bmatrix} \neq p(X_1)$$

(c) To calculate $p(\vec{x})$, we use the Markov chain property:

$$\begin{aligned} p(\vec{x}) &= p(X_1 = A_3)\ \cdot p(X_2 = A_1|\ X_1 = A_3) \cdot\ p(X_3 = A_1|\ X_2 = A_1)\ \cdot \\ &\quad p(X_4 = A_3|\ X_3 = A_1) \cdot\ p(X_5 = A_1|\ X_4 = A_3) \\ &= \frac{1}{2} \cdot \frac{1}{8} \cdot \frac{3}{4} \cdot \frac{1}{8} \cdot \frac{1}{8} \end{aligned}$$

2. (a)

$$\begin{aligned} p(X_{2j} = A_k) &= \sum_{k=1}^{3} p(\ X_{2j},\quad X_{2j-1} = A_k\ ) \\ p(X_{2j} = A_1) &= \frac{7}{24}, \\ p(X_{2j} = A_2) &= \frac{8}{24}, \\ p(X_{2j} = A_3) &= \frac{9}{24} \\ p(X_{2j-1} = A_k) &= \sum_{k=1}^{3} p(\ X_{2j} = A_k,\quad X_{2j-1}) \\ p(X_{2j-1} = A_1) &= \frac{8}{24} \\ p(X_{2j-1} = A_2) &= \frac{8}{24} \\ p(X_{2j-1} = A_3) &= \frac{8}{24} \end{aligned}$$

(b) The sequence is not stationary, since the marginal probabilities $p(X_{2j-1})$ and $p(X_{2j})$ are not the same.

(c)

$$p(X_{2j} \mid X_{2j-1}) = \frac{p(X_{2j}, X_{2j-1})}{p(X_{2j-1})}$$

And therefore:

$$P = \begin{bmatrix} \frac{1}{8} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{4} \\ \frac{3}{8} & \frac{1}{4} & \frac{1}{2} \end{bmatrix}$$

whose row i and column k hold the conditional probability

$$p(\ X_{2j} = A_i|\ \ X_{2j-1} = A_k\ )\ .$$

(d)

$$\begin{aligned} H &= \sum_i \sum_k p(\ X_{2j} = A_i, \quad X_{2j-1} = A_k\ )\ \log \frac{1}{p(\ X_{2j} = A_i, \quad X_{2j-1} = A_k\ )} \\ &= \frac{1}{3}\log 12 + \frac{1}{2}\log 6 + \frac{1}{24}\log 24 + \frac{1}{8}\log 8 \end{aligned}$$

Header files

1. (a) i. Code, C^*, for the group (naively we make it 2 bits).

   ii. Code, C^{**}, for the length of the string (use Golomb or Elias for example).

   iii. Code, C^{***}, for the actual string (use ASCII code).

   Putting this together we have something like:

$$C^*(group)C^{**}(length)C^{***}(symbol1)....C^{***}(symboln)$$

   (b) If both the encoder and the decoder knew the occurrence probabilities of the groups and each group element then Huffman codes (for example) could be constructed for the group number and each groups respective elements.

   (c) Specify how could the encode sent optimal phrase codes to the decoder as a header file.

```
for all groups j {
    specify number of bits in codeword for group j
    specify codeword for group j
    for all elements k of group j {
        specify number of bits in codeword for element k
        specify codeword for element k
    }
}
```