Student Number: ▪▪▪▪▪▪        Name: <u>Bryan Hoang</u>

,,,

6. (15 points)

(a) Answer:

Let $\mathcal{S}$ and $\mathcal{U}$ denote the set of all suffix and UD first-order binary VLCs, respectively.

First, note that since $\mathcal{S} \subseteq \mathcal{U}$, then

$$\min_{\mathcal{C} \in \mathcal{U}} \bar{L} \leq \min_{\mathcal{C} \in \mathcal{S}} \bar{L}$$
$$\Rightarrow \bar{L}_{\mathrm{UD}} \leq \bar{L}_{\mathrm{S}} \tag{1}$$

Now let $\mathcal{C}^*$, with codeword lengths $l_1^*, \ldots, l_n^*$, be the <u>optimal</u> UD code ($\mathcal{C}^* \in \mathcal{U}$):

$$\bar{L}(\mathcal{C}^*) = \bar{L}_{\mathrm{UD}}$$

where $\bar{L}(\mathcal{C})$ is the expected cost function, $\bar{L}$, of $\mathcal{C}$.

Then the codeword lengths $\{l_i^*\}_{i=1}^M$ of $\mathcal{C}^*$ must satisfy the Kraft inequality (in base D) since $\mathcal{C}^*$ is UD.

Then by the Theorem in class on the "Kraft Inequality for Prefix codes", $\exists$ prefix code $\mathcal{C}'$ with the same codeword lengths as $C^*$: $l_i' = l_i^*$, $i = 1, \ldots, M$. We can reverse the codewords of $\mathcal{C}'$ to obtain a suffix code $\mathcal{C}'' \in \mathcal{S}$, with the same codeword lengths as $C^*$: $l_i'' = l_i^*$, $i = 1, \ldots, M$.

$$\therefore \bar{L}(\mathcal{C}'') = \sum_{i=1}^M p_i c(l_i'')$$
$$= \sum_{i=1}^M p_i c(l_i^*)$$
$$= \bar{L}_{\mathrm{UD}}$$

<u>But</u>, $\bar{L}(\mathcal{C}'') \geq \bar{L}_{\mathrm{S}}$ by the definition of $\bar{L}_{\mathrm{S}}$.

$$\therefore \bar{L}_{\mathrm{UD}} \geq \bar{L}_{\mathrm{S}}. \tag{2}$$

Then by (??) and (??), we have $\boxed{\bar{L}_{\mathrm{UD}} = \bar{L}_{\mathrm{S}}}$.

(b) Answer:

<u>Compression efficiency</u> is one appropriate metric, where we want to choose a code alphabet size $D$ and maximize order $n$ so that the code's average code rate, $\overline{R}$, is as close as possible to the DMSs source entropy, $H_D(X)$.

<u>Encoding delay</u> is another metric, where we want to minimize order $n$ and design a code with minimal variance in length to minimize the delay.

<u>Storage and computational complexity</u> is a third metric, where we want to minimize the size of the source alphabet by minimizing the order $n$.

Due to these considerations, we need to balance the metrics and evluate the trade-offs between different solutions depending on the design application.