# EE5139R: Problem Set 5
# Assigned: 07/09/16, Due: 14/09/16

1. **Huffman's algorithm**: Let $p_1 > p_2 > p_3 > p_4$ be the symbol probabilities for a source alphabet size $M = |\mathcal{X}| = 4$.

   (a) What are the possible sets of codeword lengths $\{l_1, l_2, l_3, l_4\}$ for a Huffman code for the given type of source?
   **Solution**: $\{(1, 2, 3, 3), (2, 2, 2, 2)\}$ by Kraft's inequality.

   (b) Suppose that $p_1 > p_3 + p_4$. What are the possible sets of codeword lengths now?
   **Solution**: Note that $p_1 > p_3 + p_4$ means that at the second stage of the Huffman algorithm, $p_2$ will merge with the node $p_3 + p_4$ (combined at the first stage). So $l_1 = 1$ and the codeword lengths are $\{1, 2, 3, 3\}$.

   (c) What are the possible sets of codeword lengths if $p_1 < p_3 + p_4$?
   **Solution**: Similar argument for $p_1 < p_3 + p_4$. At the second stage $p_1$ will merge with $p_2$ (since they are now the two lowest probabilities). In this case, the lengths will be $\{2, 2, 2, 2\}$.

   (d) What is the smallest value of $\rho$ such that $p_1 > \rho$ implies that $l_1 = 1$ for the type of source in part (a)? (Hint: what are the largest values of $p_1, p_3, p_4$ for which part (b) holds?)
   **Solution**: Keeping in mind the assumptions $\{p_1 > p_2 > p_3 > p_4\}$, and $\sum_i p_i = 1$ and following the hint, we examine a few relevant cases. Intuitively, we see that $p_1$ would like to be as small as possible and $(p_2, p_3, p_4)$ as large as possible. We note 2 extreme cases (i) when $(p_2, p_3, p_4)$ are close to being uniformly distributed (it is easy to see in this case that if $p_1 > 2/5$ then $l_1 = 1$), and (ii) $p_2$ is almost the same as $p_1$ (it is easy to see in this case that if $p_1 > 1/3$, then $l_1 = 1$). Let's assume the worst case that $\rho = 2/5$ and show that we are correct. Claim: $\rho = 2/5$ implies $l_1 = 1$. We show by contraction, i.e. that if $p_1 > 2/5$ then $l_1 = 2$, then one of our assumptions will be violated. Suppose $l_1 = 2$, then $p_3 + p_4 > p_1 > 2/5$ which implies that $p_3 > 1/5$ (since $p_3 > p_4$) and $p_2 > 1/5$ (since $p_2 > p_3$), but $p_1 + p_3 + p_4 > p_1 > 4/5$ which implies $p_2 < 1/5$ (since $\sum_i p_i = 1$). A contradiction. So now we know that if $\rho > 2/5$, then $l_1 = 1$. But is this the lowest $\rho$? We show that for any $0 < \epsilon < 2/5$, that for $p_1 = 2/5 - \epsilon$, we can find a $(p_2, p_3, p_4)$ such that $p_1 < p_3 + p_4$ giving $l_1 = 2$. For the 2 cases we examined, note that the equiprobable case is the worst. Since probabilities are separated by strict inequalities, we cannot make the probabilities equal to we tweak the probabilities by a small amount. Choose $p_2 = 1/5 + \epsilon/6$, $p_3 = 1/5 + 2\epsilon/6$ and $p_4 = 1/5 + 3\epsilon/6$. Then $p_1 < p_3 + p_4$. Thus, we have shown that indeed $\rho = 2/5$ is the minimum value to guarantee that $l_1 = 1$.

   (e) What is the largest value of $\lambda$ such that $p_1 < \lambda$ implies $l_1 = 2$?
   **Solution**: Similar to the argument above. Easy to see for the extreme cases examine above that for case (i) if $p_1 < 2/5$, then $l_1 = 2$ and for case (ii) that if $p_1 < 1/3$ implies $l_1 = 2$. Assume the worst case that $\lambda = 1/3$ and we show is correct. Claim: $p_1 < 1/3$ implies $l_1 = 2$. Suppose that $l_1 = 1$, then $p_3 + p_4 < p_1 < 1/3$ which implies that $p_1 + p_3 + p_4 < 2/3$ which implies that $p_2 > 1/3$. but $p_2 < p_1 < 1/3$ a contradiction. Is this the largest $\lambda$? Again we show by construction that for any $0 < \epsilon < 1/6$, and for any $p_1 = 1/3 + \epsilon$, we can find a $(p_2, p_3, p_4)$ such that $l_1 = 1$. From our toy example at the beginning, we want $p_2$ to be close to $p_1$. Choose $p_2 = 1/3 + \epsilon/2$, $p_3 = 1/6 - \epsilon$ and $p_4 = 1/6 - \epsilon/2$. Then $p_1 > p_3 + p_4$ and $l_1 = 1$.

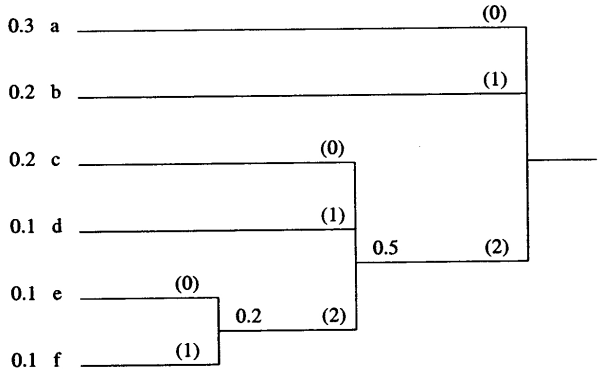


Figure 1: Ternary Huffman Code Tree

2. (**Optional**): **Ternary Huffman Codes**: Consider extending the Huffman procedure to codes with ternary symbols $\{0, 1, 2\}$. Think in terms of codewords as leaves of ternary trees. Assume an alphabet with $M = 4$ symbols. Note that you cannot draw a full ternary tree with 4 leaves.

   (a) By starting with a tree of 3 leaves and extending the tree by converting leaves into intermediate nodes, show for what values of $M$ it is possible to have a complete ternary tree.
   **Solution**: Grow a full ternary tree at each step. The smallest tree has 3 leaves. For the next largest full tree, convert one of the leaves into an intermediate node and grow 3 leave from that node. We lose 1 leaf but gain 2 more at each growth extension. Thus, $M = 3 + 2n$ where $n$ is an integer.

   (b) Explain how to generalize the Huffman procedure to ternary symbols bearing in mind your result in part (a).
   **Solution**: It is clear that for optimality, all the unused leaves in the tree must have the same length as the longest codeword. For $M$ even, combine the 2 lowest probabilities into a node at the first step, then combine the 3 lowest probability nodes for the rest of the steps until the root node. If $M$ is odd, a full ternary tree is possible, so combine the 3 lowest probability nodes at each step.

   (c) Use your algorithm for the set of probabilities $\{0.3, 0.2, 0.2, 0.1, 0.1, 0.1\}$.
   **Solution**: The ternary Huffman code is $\{a \to 0, b \to 1, c \to 20, d \to 21, e \to 220, f \to 221\}$. See Fig. 1

3. (**Optional**): **Huffman's algorithm**: A source with an alphabet size of $M = |\mathcal{X}| = 4$ has symbol probabilities $\{1/3, 1/3, 2/9, 1/9\}$.

   (a) Use the Huffman algorithm to find an optimal prefix-free code for this source.
   **Solution**: One optimal PF code from the Huffman algorithm is $\{a \to 00, b \to 01, c \to 10, d \to 11\}$. See Fig. 2.

   (b) Use the Huffman algorithm to find another optimal prefix-free code with a different set of lengths.
   **Solution**: Another optimal PF code from the Huffman algorithm is $\{a \to 0, b \to 10, c \to 110, d \to 111\}$. See Fig. 3.

   (c) Find another prefix-free code that is optimal but cannot result from using the Huffman algorithm.
   **Solution**: One example of an optimal PF code is $\{a \to 00, b \to 10, c \to 01, d \to 11\}$. This code cannot result from using the Huffman algorithm since the two lowest siblings $\{c, d\}$ are not siblings in the code tree.

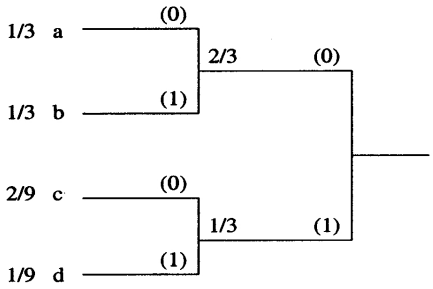


Figure 2: Huffman Code 1

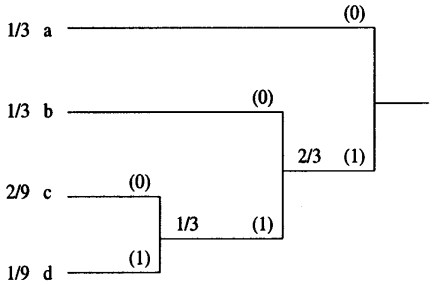


Figure 3: Huffman Code 2

4. Consider a binary prefix-free (PF) code with codeword lengths

$$l_1 \leq l_2 \leq \ldots \leq l_M.$$

We construct this PF randomly as follows: For each $k \in \{1, 2, \ldots, M\}$ the codeword $\mathcal{C}(k)$ of length $l_k$ is chosen independently from the set of all $2^{l_k}$ possible binary strings with length $l_k$ according to the uniform distribution. Let $P_M(\texttt{good})$ be the probability that the so-constructed code is PF.

(a) (2 points] Consider a source with a binary alphabet so $M = 2$ and there are only 2 lengths $l_1 \leq l_2$. Show that

$$P_2(\texttt{good}) = \left(1 - 2^{-l_1}\right)^+$$

where

$$(x)^+ = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases}.$$

**Solution**: Suppose the first codeword is $\mathcal{C}(1) = b_1, b_2, \ldots, b_{l_1}$ of length $l_1$. To ensure that $\mathcal{C}$ is PF, the second codeword $\mathcal{C}(2)$ can't have prefix $b_1, b_2, \ldots, b_{l_1}$. This occurs with probability $(1-2^{-l_1})^+$. Since $b_1, b_2, \ldots, b_{l_1}$ is chosen arbitrarily and uniformly, we have established the claim.

(b) (3 points) Prove by induction on $M$ that

$$P_M(\texttt{good}) = \prod_{k=1}^{M} \left(1 - \sum_{j=1}^{k-1} 2^{-l_j}\right)^+.$$

**Solution**: Assume claim is true for $K$ so

$$P_K(\texttt{good}) = \prod_{k=1}^{K} \left(1 - \sum_{j=1}^{k-1} 2^{-l_j}\right)^+ .$$

Now, the codewords $\mathcal{C}(1), \mathcal{C}(2), \ldots, \mathcal{C}(K)$ have been chosen. Hence, the $(K+1)$'s codeword $\mathcal{C}(K+1)$ must not contain any of the prefixes $\mathcal{C}(1), \mathcal{C}(2), \ldots, \mathcal{C}(K)$. This occurs with probability $1 - \sum_{j=1}^{K} 2^{-l_j}$ because each of the lengths of $\mathcal{C}(j)$ is $l_j$. Since the codeword selection is independent, we have

$$P_{K+1}(\texttt{good}) = P_K(\texttt{good}) \left(1 - \sum_{j=1}^{K} 2^{-l_j}\right)$$

which establishes the claim.

(c) (1 points) Is the following statement true or false. Provide a brief reason.

| $P_M(\texttt{good}) > 0$ if and only if there exists a PF code for a source with alphabet size $M$ |
|---|

**Solution**: Yes. The statement is true. By the random coding argument.

(d) (4 points) Use the above parts to prove Kraft's inequality.

**Solution**: This is rather tricky. If $P_M(\texttt{good}) > 0$ it means that all the terms in the product $(1 - \sum_{j=1}^{M-1} 2^{-l_j})^+ > 0$. This means that

$$\sum_{j=1}^{M-1} 2^{-l_j} < 1.$$

This means that

$$\sum_{j=1}^{M-1} 2^{-l_j + l_M} < 2^{l_M}$$

Since $l_M \geq l_j$ for all $j = 1, \ldots, M$, both sides are integers, we must have

$$\left(\sum_{j=1}^{M-1} 2^{-l_j + l_M}\right) + 1 \leq 2^{l_M}.$$

Now dividing both sides by $2^{l_M}$, we have

$$\left(\sum_{j=1}^{M-1} 2^{-l_j}\right) + 2^{-l_M} \leq 1,$$

which is exactly Kraft's inequality.

5. (**Optional**):

**Variable-Length Prefix-Free Codes**: A discrete memoryless source emits iid random variables $X_1, X_2, \ldots$. Each random variable $X$ has the symbols $\{a, b, c\}$ with probabilities $\{0.1, 0.4, 0.5\}$, respectively.

(a) Find the expected length $\bar{L}_{\min}$ of the best variable-length prefix-free code for $X$.

**Solution**: The best one-to-variable PF code can be found using Huffman coding. Fig. 4 illustrates the procedure which yields the following mapping $\{a \to 11, b \to 10, c \to 0\}$. Thus, the length of the code $\bar{L}_{\min} = 0.5 \times 1 + 0.4 \times 2 + 0.1 \times 2 = 1.5$ bits/symbol.

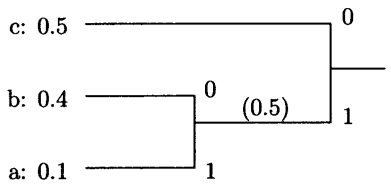


Figure 4: One-to-variable Huffman-Coding

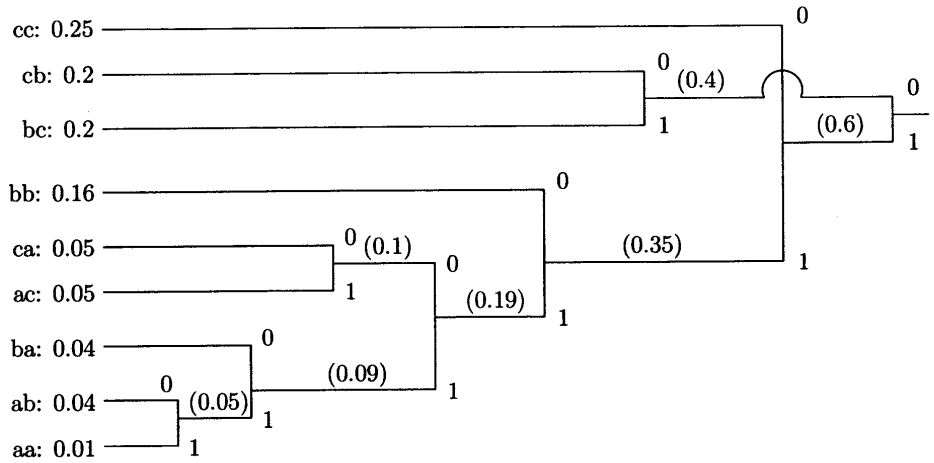


Figure 5: Two-to-variable Huffman-Coding

(b) Find the expected length $\bar{L}_{\min,2}$ normalized to bits per symbol, of the best variable-length prefix-free code for $X^2 = (X_1, X_2)$.
**Solution**: Map each pair of source alphabets to a codeword using Huffman coding. The mapping is $\{aa \to 111111, ab \to 111110, ac \to 11101, ba \to 11110, bb \to 110, bc \to 01, ca \to 11100, cb \to 00, cc \to 10\}$. See Fig. 5. Thus, $\bar{L}_{\min,2} = 1.39$ bits/symbol.

(c) Is it true that for any DMS, $\bar{L}_{\min} \geq \bar{L}_{\min,2}$? Explain.
**Solution**: Construct a two-to-variable length code[1] say $\mathcal{C}_2$, thus simply map the source letter pair $(X_1, X_2)$ to $(\mathcal{C}(X_1), \mathcal{C}(X_2))$ where $\mathcal{C}$ is the optimal one-to-variable length code. Clearly, the resulting expected number of bits/source symbol is the same as that for the code $\mathcal{C}$. Hence, we have constructed a two-to-variable length code that does no worse than $\mathcal{C}$. By definition the *optimal* two-to-variable length code can only do better than $\mathcal{C}_2$. Hence, for any DMS

$$\bar{L}_{\min} = \bar{L}_2 \geq \bar{L}_{\min,2}$$

where $\bar{L}_2$ is the expected bits per source symbol for the trivial code $\mathcal{C}_2$ that we constructed.

6. **(Optional) From 2013/2014 Final Exam** Consider a discrete memoryless source $X$ with alphabet $\{1, 2, \ldots, M\}$. Suppose that the symbol probabilities are ordered and satisfy $p_1 > p_2 > \ldots > p_M$ and also satisfy $p_1 < p_{M-1} + p_M$. Let $l_1, l_2, \ldots, l_M$ be the lengths of a prefix-free code of minimum expected length for such a source.

[1] A code that encodes two source symbols at a time.

(a) **(1 point) TRUE** or **FALSE**: $l_1 \leq l_2 \leq \ldots \leq l_M$.

**TRUE.** *This was done in class. For an optimal code, if $p_i > p_j$, then $l_i \leq l_j$. Suppose otherwise, i.e., that $l_i > l_j$, then I can create a new code with lengths $l'_i = l_j$ and $l'_j = l_i$ for symbols $i$ and $j$ respectively. It is then easy to see that the expected cost will decrease. Indeed, the contribution to the expected cost from symbols $i$ and $j$ in the old code is $c_{ij}^{\text{old}} = p_i l_i + p_j l_j$. The contribution to the expected cost from symbols $i$ and $j$ in the new code is $c_{ij}^{\text{new}} = p_i l_j + p_j l_i$. Because $p_i > p_j$ and $l_i > l_j$, we have that $c_{ij}^{\text{old}} > c_{ij}^{\text{new}}$ which is a contradiction and so the initial assumption that $l_i > l_j$ must be false. Hence, $l_i \leq l_j$. This argument can be repeated for all symbols.*

(b) **(2 points)** Show that if the Huffman algorithm is used to generate the above code, then $l_M \leq l_1 + 1$. (*Hint: You may use part (a).*)

*For the Huffman algorithm, we will first merge symbols $M-1$ and $M$ to form a symbol with probability $p_{M-1} + p_M$. The codeword for this new symbol is $l_M - 1$. Since $p_{M-1} + p_M > p_1$, by part (a), we must have that $l_1 \geq l_M - 1$ which is the desired result.*

(c) **(Bonus 2 points)** Show that $l_M \leq l_1 + 1$ for *any* (not necessarily Huffman generated) prefix-free code of minimum expected length. (*Hint: A minimum-expected-length code must be full.*)

*A minimum-expected-length code must be full, and thus the codeword for letter $M$ must have a sibling, say letter $j$. Since $p_j \geq p_{M-1}$, we have $p_j + p_M > p_1$ because of the condition $p_1 < p_{M-1} + p_M$. Let $l'$ be the length of the intermediate node that is parent to $j$ and $M$. Now $l' \leq l_1$ since otherwise the codeword for 1 could be interchanged with this intermediate node for a reduction in $\bar{L}$. Again $l_M - 1 \leq l_1$.*

(d) **(2 points)** Suppose $M = 2^k$ for some integer $k$. Show that all codewords must have the same length. (*Hint: What does the Kraft inequality look like for an optimal code? Consider the three cases $l_1 = k$, $l_1 < k$ and $l_1 > k$.*)

*An optimal prefix-free code must be full. For full codes, Kraft must be satisfied with equality, i.e., $\sum_{j=1}^{2^k} 2^{-l_j} = 1$. First assume that $l_1 = k$. Then all codewords have length $k$ or $k+1$, but as mentioned above, the Kraft's inequality can be satisfied with equality (i.e., the code can be full) only if all codewords have length $k$. If $l_1 > k$, then $l_j > k$ for all $j$ and the Kraft inequality can not be satisfied with equality. Finally, if $l_1 < k$ with $l_j \leq k$, then the Kraft inequality can not be met. Thus all codewords have length $k$.*

7. (**Optional**): **From 2013/2014 Final Exam:** Consider a discrete memoryless source $X$ with alphabet $\{1, 2, \ldots, M\}$ with probabilities $p_1 \geq p_2 \geq \ldots \geq p_{M-1} > 0$. The Huffman algorithm operates by joining the two least likely symbols together as siblings and then constructs an optimal prefix-free code for a reduced source $X'$ in which the symbols of probability $p_M$ and $p_{M-1}$ have been replaced by a single symbol of probability $p_M + p_{M-1}$. The expected code-length $\bar{L}$ of the code for the original source $X$ is then equal to $\bar{L}' + p_M + p_{M-1}$ where $\bar{L}'$ is the expected code-length of $X'$. The entropy of $X$ is defined as

$$H(X) = \sum_{j=1}^{M} p_j \log p_j.$$

(e) **(2 points)** Express the entropy $H(X)$ for the original source in terms of the entropy $H(X')$ of the reduced source as

$$H(X) = H(X') + (p_M + p_{M-1}) h(\gamma)$$

where $h(\gamma) = -\gamma \log \gamma - (1-\gamma) \log(1-\gamma)$. Find the required $\gamma$.

*We have*

$$H(X) = \sum_{i=1}^{M} -p_i \log p_i, \qquad H(X') = \sum_{i=1}^{M-2} -p_i \log p_i - (p_{M-1} + p_M) \log(p_{M-1} + p_M).$$

*Subtracting we obtain*

$$\begin{aligned} H(X) - H(X') &= (p_{M-1} + p_M) \log(p_{M-1} + p_M) - p_M \log p_M - p_{M-1} \log p_{M-1} \\ &= (p_{M-1} + p_M) h\left(\frac{p_M}{p_M + p_{M-1}}\right). \end{aligned}$$

*Hence* $\gamma = \frac{p_M}{p_M + p_{M-1}}$.

(f) **(1 point)** In the code tree generated by the Huffman algorithm, let $v_1$ denote the intermediate node that is the parent of the leaf nodes for symbols $M$ and $M-1$. Let $q_1 = p_M + p_{M-1}$ be the probability of reaching $v_1$ in the code tree. Similarly, let $v_2, v_3, \ldots$ denote the subsequent intermediate nodes generated by the Huffman algorithm. How many intermediate nodes are there, including the root node of the entire tree?

*Each step of the Huffman algorithm reduces the number of symbols by 1 until only 1 node (the root) is left. Thus there are $M-1$ intermediate nodes, counting the root.*

(g) **(2 points)** Let $q_1, q_2, \ldots$ be the probabilities of reaching the intermediate nodes $v_1, v_2, \ldots$ (note that the probability of reaching the root node is 1). Show that $\bar{L} = \sum_{i=1}^{M-1} q_i$. (*Hint: Note that $\bar{L} = \bar{L}' + q_1$. Why?*)

*We showed in class that the Huffman algorithm is optimal. Thus at every step, optimality is retained, i.e., that $\bar{L} = \bar{L}' + q_1$ where $\bar{L}$ is the optimal expected length for the PF code for $X$ and $\bar{L}'$ is the optimal expected length for the PF code for $X'$. After the second step of the algorithm, $\bar{L}'$ is related to the minimum expected length, say $\bar{L}^{(2)}$ of the further reduced code by $\bar{L}' = \bar{L}^{(2)} + q_2$. Thus, $\bar{L} = \bar{L}^{(2)} + q_1 + q_2$. Proceeding to step $M-1$, (or more formally using induction) we have $\bar{L} = \bar{L}^{(M-1)} + q_1 + q_2 + \ldots q_{M-1}$. Since the expected length of the root node $\bar{L}^{(M-1)} = 0$, $\bar{L} = \sum_{i=1}^{M-1} q_i$.*

8. **2014/15 Midterm: Huffman for Large Alphabets: Huffman for Large Alphabets**:

(a) (3 points) Find the codeword lengths of an optimal binary encoding (Huffman code) of the distribution

$$P_X(i) = \frac{1}{100}, \qquad \text{for } i = 1, 2, \ldots, 100.$$

That is the source is uniform on 100 symbols. Just find the optimal lengths (how many codewords of various lengths). You do not need to produce the actual code.

*You do not need to use a calculator but the following is useful:* $6 \le \log_2 100 \le 7$.

**Solution**: Since the distribution is uniform, the Huffman tree will consist of word lengths of $\lceil \log_2 100 \rceil = 7$ and $\lfloor \log_2 100 \rfloor = 6$. There are 64 nodes of depth 6, of which $(64 - k)$ will be leaf nodes; and there are $k$ nodes of depth 6 which will form $2k$ leaf nodes of depth 7. Since the total number of leaf nodes is 100, we have

$$(64 - k) + 2k = 100, \qquad \Rightarrow \qquad k = 36.$$

So there are $64 - 36 = 28$ codewords of length 6 and $2 \times 36 = 72$ codewords of length 7.