Maximal Codeword Lengths in Huffman Codes

Y. S. Abu-Mostafa
California Institute of Technology, Electrical Engineering Department

R. J. McEliece
Communications Systems Research Section

In this article, the authors consider the following question about Huffman coding, which is an important technique for compressing data from a discrete source. If p *is the smallest source probability, how long, in terms of* p, *can the longest Huffman codeword be? It is shown that if* p *is in the range* $0 < p \leq 1/2$, *and if* K *is the unique index such that* $1/F_{K+3} < p \leq 1/F_{K+2}$, *where* F_K *denotes the* K*th Fibonacci number, then the longest Huffman codeword for a source whose least probability is* p *is at most* K, *and no better bound is possible. Asymptotically, this implies the surprising fact that for small values of* p, *a Huffman code's longest codeword can be as much as 44 percent larger than that of the corresponding Shannon code.*

I. Introduction and Summary

Huffman coding is optimal (in the sense of minimizing average codword length) for any discrete memoryless source, and Huffman codes are used widely in data compression applications. In many situations it would be useful to have an easy way to estimate the longest Huffman codeword length for a given source, without having to go through Huffman's algorithm, but since there is no known closed-form expression for the Huffman codeword lengths, no such estimate immediately suggests itself. However, since the longest codeword will always be associated with the least-probable source symbol, one way to address this problem is to ask the following question: If p is the smallest source probability, how long, in terms of p, can the longest Huffman codeword be? It turns out that this quantity, denoted by $L(p)$, is easy to calculate, and so $L(p)$ provides an "easy estimate" of the longest Huffman codeword length.

The formula for $L(p)$ involves the famous Fibonacci numbers $(F_n)_{n\geq 0}$, which are defined recursively, as follows:

$$F_0 = 0, F_1 = 1, \text{ and } F_n = F_{n-1} + F_{n-2} \text{ for } n \geq 2 \quad (1)$$

Thus, $F_2 = 1$, $F_3 = 2$, $F_4 = 3$, $F_5 = 5$, $F_6 = 8$, etc. The Fibonacci numbers and their properties are discussed in detail in [1, Section 1.2.8]. Here is the main result of this article. (Note that since the definition of $L(p)$ assumes p to be the smallest probability in a source, p must lie in the range $0 < p \leq 1/2$.)

Theorem 1. Let p be a probability in the range $0 < p \leq 1/2$, and let K be the unique index such that

$$\frac{1}{F_{K+3}} < p \leq \frac{1}{F_{K+2}} \tag{2}$$

Then $L(p) = K$. Thus $p \in (1/3, 1/2]$ implies $L(p) = 1$, $p \in (1/5, 1/3]$ impies $L(p) = 2$, $p \in (1/8, 1/5]$ implies $L(p) = 3$, etc.

It is easy to prove by induction that the Fibonacci numbers satisfy the following inequalities:

$$\phi^{n-2} < F_n < \phi^{n-1} \qquad \text{for } n \geq 3 \tag{3}$$

where $\phi = (1+\sqrt{5})/2 = 1.618\ldots$ is the "golden ratio." By combining inequality (3) with Theorem 1, one sees that

$$\log_\phi \frac{1}{p} - 2 < L(p) < \log_\phi \frac{1}{p} \tag{4}$$

which, in turn, implies that

$$\lim_{p \to 0} \frac{L(p)}{\log_\phi \frac{1}{p}} = 1 \tag{5}$$

Since $\log_\phi x = (\log_2 x)/(\log_2 \phi) = 1.4404 \log_2 x$, Eq. (5) implies the surprising fact that for small values of p, a Huffman code's longest codeword can be as much as 44 percent larger than that of the corresponding (in general, suboptimal) Shannon code [2, Chapter 5], which assigns a symbol with probability p a codeword of length $\lceil \log_2 \frac{1}{p} \rceil$.

Theorem 1 is closely related to a result of Katona and Nemetz [4], which identifies the length of the longest possible Huffman codeword for a source symbol of probability p (whether or not p is the smallest source probability). Denoting this quantity by $L^*(p)$, their result is as follows:

Theorem 2. (Katona and Nemetz [4]) Let p be a probability in the range $0 < p < 1$, and let K be the unique index such that

$$\frac{1}{F_{K+2}} \leq p < \frac{1}{F_{K+1}} \tag{6}$$

Then $L^*(p) = K$. Thus, $p \in [1/2, 1)$ implies $L^*(p) = 1$, $p \in [1/3, 1/2)$ implies $L^*(p) = 2$, $p \in [1/5, 1/3)$ implies $L^*(p) = 3$, etc.

By comparing Theorems 1 and 2, one sees that $L^*(p) = L(p) + 1$ unless p is the reciprocal of a Fibonacci number, in which case $L^*(p) = L(p)$.[1]

II. Proof of Theorem 1

The proof of Theorem 1 is in two parts. First, it will be shown that if $p > 1/F_{K+3}$, then in any Huffman code for a source whose smallest probability is p, the longest codeword length is at most K. In fact, a considerably stronger result will be proved. The class of efficient prefix codes will be defined, and it will be shown that any Huffman code, and in fact any optimal code for a given source, is efficient. Then it will be shown that if $p > 1/F_{K+3}$, in any efficient code for a source whose smallest probability is p, the longest codeword length is at most K. In the second half of the proof, it will be shown that if $p \leq 1/F_{K+2}$, there exists a source whose smallest probability is p, which has at least one Huffman code whose longest word has length K. As an extension, it will be seen that if $p < 1/F_{K+2}$, there exists a source whose smallest probability is p, and for which *every* optimal code has the longest word of length K. (If $p = 1/F_{K+2}$, however, there is no such source.)

Now comes the definition of efficient prefix codes, which is best stated in terms of the associated binary code tree (see Fig. 1). Each source symbol and its corresponding codeword is associated with a unique terminal node on the tree. Also, each node in the tree is assigned a probability. The probability of a terminal node is defined to be the probability of the corresponding source symbol, and the probability of any other node of the code tree is defined to be the sum of the probabilities of its two "children." The *level* of the root node is defined to be zero, and the level of every other node is defined to be one more than the level of its parent. Two nodes descended from the same parent node are called *siblings*. Figure 1 shows two different code trees for the source $[3/20, 3/20, 3/20, 3/20, 8/20]$. The tree in Fig. 1(a) corresponds to the prefix code $\{000, 001, 01, 10, 11\}$, and the tree in Fig. 1(b) corresponds to $\{000, 001, 010, 011, 1\}$.

Definition. A prefix code for a source S is efficient if every node except the root in the code tree has a sibling, and if level(v) < level(v') implies $p(v) \geq p(v')$.

[1] In fact, however, if one were to make a subtle change in the definition of $L(p)$, this special case would disappear. The change required is to define $L(p)$ as the minimum maximum Huffman codeword length over all Huffman codes for a source with p as the least probability, where the outer minimum is over all Huffman codes for a given source.

Gallager [3] noted that every Huffman tree is efficient, but in fact it is easy to see more generally that every optimal tree is efficient. This is because in an *inefficient* tree, with nodes v and v' such that level(v) < level(v') but $p(v) < p(v')$, by interchanging the subtrees rooted at v and v', one arrives at a new code tree for the same source, whose average length has been reduced by exactly (level(v') − level(v))$(p(v') - p(v))$. However, it is not true that every efficient code is optimal. Indeed, Fig. 1 shows two different efficient code trees for the source $[3/20, 3/20, 3/20, 3/20, 8/20]$. The code in Fig. 1(b) is optimal, but the one in Fig. 1(a) is not.

Theorem 3. If $p > 1/F_{K+3}$, then in any efficient prefix code for a source whose least probability is p, the longest codeword length is at most K.

Proof: The contrapositive will be proved, i.e., if p is the least probability in a source that has an efficient prefix code whose longest word has length $\geq K+1$, then $p \leq 1/F_{K+3}$.

Thus, suppose that S is a source whose least probability is p and that there is an efficient prefix code for S whose longest word is of length $\geq K+1$. In the code tree for this code, there must be a path of length $K+1$ starting from the terminal node, which corresponds to the longest word and moves upward toward the root. This path is shown in Fig. 2 as the path whose probabilities are $p_0, p_1, \ldots, p_{K+1}$. Since the code is assumed to be efficient, each of the vertices in this path (except possibly the top vertex) has a sibling; these siblings are shown in Fig. 2 as having probabilities $q_0, q_1, \ldots, q_K$. Now one can prove the following:

$$p_i \geq F_{i+2}p \qquad \text{for } i = 0, 1, \ldots, K+1 \tag{7}$$

The proof of (7) is by induction. For $i = 0$, (7) merely says that $p_0 \geq p$, which is true since $p_0 = p$, by definition. Also, note that $q_0 \geq p$ since p is the least source probability. Thus, $p_1 = p_0 + q_0 \geq p + p = 2p = F_3 p$, which proves (7) for $i = 1$. For $i \geq 2$, one has $p_i = p_{i-1} + q_{i-1}$. But $p_{i-1} \geq F_{i+1}p$ by induction, and $q_{i-1} \geq p_{i-2}$ since the code is efficient (q_{i-1} is a higher level node than p_{i-2}). Thus, one has $q_{i-1} \geq p_{i-2} \geq F_i p$ by induction, and so $p_i = p_{i-1} + q_{i-1} \geq (F_{i+1} + F_i)p = F_{i+2}p$, which completes the proof of (7).

Now consider the probability p_{K+1}. On one hand, $p_{K+1} \leq 1$; but on the other hand, $p_{K+1} \geq F_{K+3}p$, by (7). Thus, $p \leq 1/F_{K+3}$, which completes the proof. ❐

Theorem 4. If $p \leq 1/F_{K+2}$, there exists a source whose smallest probability is p and which has a Huffman code whose longest word has length K. If $p < 1/F_{K+2}$, there exists such a source for which every optimal code has a longest word of length K.

Proof: Consider the following set of $K+1$ source probabilities:

$$\left[p, \frac{F_1}{F_{K+2}}, \frac{F_2}{F_{K+2}}, \ldots, \frac{F_{K-1}}{F_{K+2}}, \frac{F_K + 1}{F_{K+2}} - p\right] \tag{8}$$

Note that p is the minimal probability for this source, since $p \leq 1/F_{K+2} = F_1/F_{K+2}$. Now, consider the code tree for this source depicted in Fig. 3, which assigns the source probability p a word of length K. This tree is in fact a Huffman tree for these probabilities, i.e., a code tree that arises when Huffman's algorithm is applied to the source of (8). To see this, one first proves that the internal vertex probabilities p_i in Fig. 3 are given by the following formula:

$$p_i = F_{i+2}/F_{K+2} - h, \quad \text{for } i = 0, 1, \ldots, K-1 \tag{9}$$

$$p_K = 1 \tag{10}$$

where $h = 1/F_{K+2} - p$.

To prove (9), one uses induction. For $i = 0$, by definition, $p_0 = p = 1/F_{K+2} - h = F_2/F_{K+1} - h$. For $i \geq 1$, one then has $p_i = p_{i-1} + F_i/F_{K+2} = (F_{i+1}/F_{K+1} - h) + F_i/F_{K+2} = F_{i+2}/F_{K+2} - h$. To prove (10), note that $p_K = p_{K-1} + (F_K + 1)/F_{K+2} - p$. But from (9), $p_{K-1} = (F_{K+1}/F_{K+2} - h)$, so that $p_K = (F_{K+1}/F_{K+2} - h) + (F_K/F_{K+2} + h) = F_{K+2}/F_{K+2} = 1$. Thus the probabilities in (8) sum to one.

It now follows that the tree in Fig. 3 is a Huffman tree, for from (9) one sees that at the ith stage ($i = 0, \ldots, K-1$), the "collapsed" source consists of the probabilities

$$[F_{i+2}/F_{K+2} - h, F_{i+1}/F_{K+2}, F_{i+2}/F_{K+2}, \ldots, F_{K-1}/F_{K+2}, F_K/F_{K+2} + h] \tag{11}$$

Plainly the two leftmost probabilities in (11), namely $F_{i+2}/F_{K+2} - h$ and F_{i+1}/F_{K+2}, are two of the smallest probabilities, and so the tree of Fig. 3 is a Huffman tree, as asserted.

Finally, note that if $h > 0$, i.e., if $p < 1/F_{K+2}$, that the leftmost two probabilities in (11) are ***uniquely*** the two

smallest probabilities in the list, so that the Huffman tree in Fig. 3 is the unique Huffman tree for the source of Eq. (8). And since the set of codeword lengths in any optimal code is the same as the set of lengths in some Huffman code, the last statement in Theorem 4 follows. ❐

By combining Theorems 3 and 4, one obtains a result that is stronger than Theorem 1.

Example 1: Let $p = 2^{-8}$. Then $1/F_{14} = 1/377 < p < 1/F_{13} = 1/233$, and so by Theorem 1, $L(2^{-8}) = 11$. More concretely, Theorem 3 shows that no Huffman code for a source whose smallest probability is 2^{-8} can have a codeword whose length is longer than 11. By Theorem 4, on the other hand, every optimal code for the source

$$\left[2^{-8}, \frac{1}{233}, \frac{1}{233}, \frac{2}{233}, \frac{3}{233}, \frac{5}{233}, \frac{8}{233}, \frac{13}{233}, \frac{21}{233}, \frac{34}{233}, \frac{55}{233}, \frac{90}{233} - 2^{-8}\right] \tag{12}$$

has a longest word of length 11. ❐

III. Extension of the Katona-Nemetz Theorem

In this section, two theorems are stated without proof. When taken together, they yield a result that is slightly stronger than Katona and Nemetz's Theorem 2. The proofs are entirely similar to the proofs of Theorems 3 and 4.

Theorem 5. Let S be a source containing a symbol a whose probability is p. If $p \geq 1/F_{K+2}$, then in any efficient prefix code for S, the length of the codeword assigned to the symbol a is at most K.

Theorem 6. Let $p < 1/F_{K+1}$. Then there exists a source S containing a symbol a whose probability is p, and such that every optimal code for S assigns a a codeword of length K. Explicitly, one such source is given by

$$S = \left[\frac{1}{F_{K+1}} - p - \epsilon, p, \frac{F_1}{F_{K+1}}, \frac{F_2}{F_{K+1}}, \ldots, \frac{F_{K-1}}{F_{K+1}} + \epsilon\right] \tag{13}$$

where ϵ is any real number such that $0 < \epsilon < 1/F_{K+2} - p$.

Example 2: Let $p = 2^{-8}$. Then $1/F_{14} = 1/377 < p < 1/F_{13} = 1/233$, and so by Theorem 2, $L^*(2^{-8}) = 12$. Indeed, by Theorem 6, every optimal code for the source

$$\left[\frac{1}{233} - 2^{-8} - \epsilon, 2^{-8}, \frac{1}{233}, \frac{1}{233}, \frac{2}{233}, \frac{3}{233}, \frac{5}{233}, \frac{8}{233}, \frac{13}{233}, \frac{21}{233}, \frac{34}{233}, \frac{55}{233}, \frac{89}{233} + \epsilon\right] \tag{14}$$

where $0 < \epsilon < 1/233 - 1/256$, assigns the symbol with probability 2^{-8} a codeword of length 12. ❐

Acknowledgment

The authors are grateful to Douglas Whiting of STAC, Inc., for suggesting this problem.

References

[1] D. E. Knuth, *The Art of Computer Programming, vol. 1: Fundamental Algorithms*, 2nd ed., Reading, Massachusetts: Addison-Wesley, 1973.

[2] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, New York: Wiley & Sons, 1991.

[3] R. V. Gallager, "Variations on a theme by Huffman," *IEEE Trans. Inform. Theory*, vol. IT-24, pp. 668–674, November 1978.

[4] G. O. H. Katona and T. O. H. Nemetz, "Huffman Codes and Self-Information," *IEEE Trans. Inform. Theory*, vol. IT-22, pp. 337–340, May 1976.

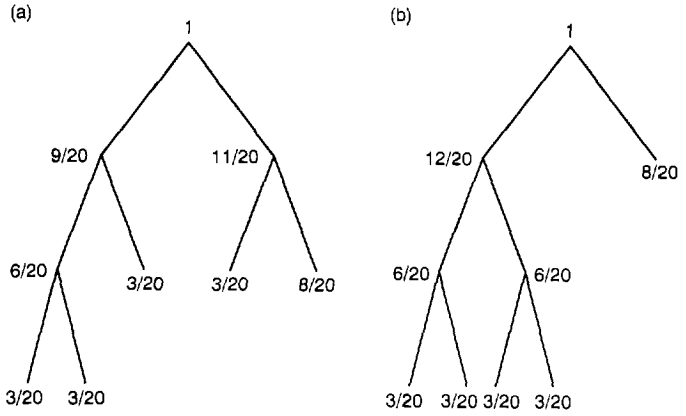

Fig. 1. Two code trees for the source [3/20, 3/20, 3/20, 3/20, 3/20]: (a) a tree that is efficient but not optimal (average length = 2.3) and (b) a tree that is optimal (average length = 2.2).

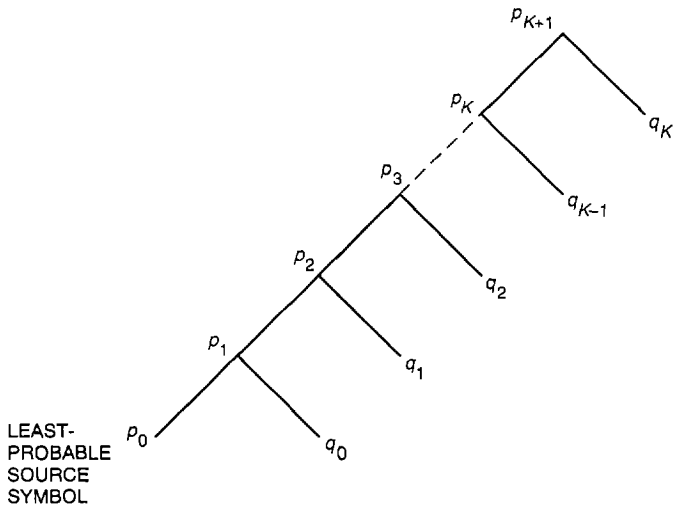

Fig. 2. A portion of an efficient code tree, in which the longest codeword has length $\geq K + 1$. p_0 is the least source probability.

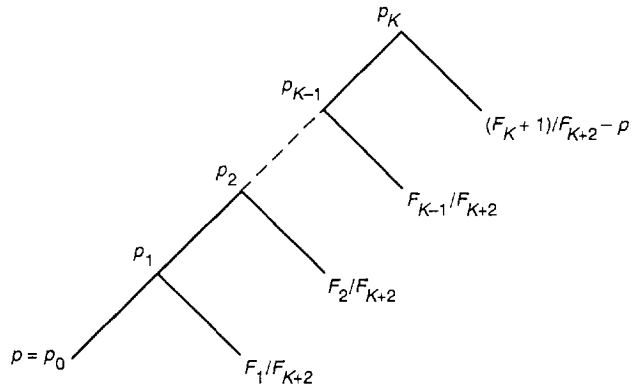

Fig. 3. A Huffman code tree for the source in (8). Its smallest probability is p, where $p \leq 1/F_{K+2}$, and its longest codeword length is K.