Improved Bounds on the Inefficiency of Length-Restricted Prefix Codes

Ruy Luiz Milidiú
milidiu@inf.puc-rio.br
Eduardo Sany Laber
laber@lmf-di.puc-rio.br

PUC-RioInf.MCC33/97 September, 1997

Abstract: Consider an alphabet $\Sigma = \{a_1, \ldots, a_n\}$ with corresponding symbol probabilities $p_1, \ldots, p_n$. The $L-$restricted prefix code is a prefix code where all the code lengths are not greater than L. The value L is a given integer such that $L \geq \lceil \log n \rceil$.

Define the average code length difference by $\epsilon = \sum_{i=1}^{n} p_i.\bar{l}_i - \sum_{i=1}^{n} p_i.l_i$, where $\bar{l}_1, \ldots, \bar{l}_n$ are the code lengths of the optimal L-restricted prefix code for Σ and $l_1, \ldots, l_n$ are the code lengths of the optimal prefix code for Σ.

Let ψ be the golden ratio 1,618. In this paper, we show that $\epsilon < 1/\psi^{L-\lceil \log(n+\lceil \log n \rceil - L) \rceil - 1}$ when $L > \lceil \log n \rceil$. We also prove the sharp bound $\epsilon < \lceil \log n \rceil - 1$, when $L = \lceil \log n \rceil$. By showing the lower bound $\frac{1}{\psi^{L-\lceil \log n \rceil + 2 + \lceil \log \frac{n}{n-L} \rceil} - 1}$ on the maximum value of ϵ, we guarantee that our bound is asymptotically tight in the range $\lceil \log n \rceil < L \leq n/2$.

Furthermore, we present an $O(n)$ time and space $1/\psi^{L-\lceil \log(n+\lceil \log n \rceil - L) \rceil - 1}$-approximative algorithm to construct L-restricted prefix codes, considering that the given probabilities are already sorted. The results presented in this paper indicate that one can efficiently implement prefix codes with length restriction, obtaining also very effective codes.

Keywords: Prefix Codes, Huffman Trees, Approximative Algorithms

Resumo: Considere o alfabeto $\Sigma = \{a_1, \ldots, a_n\}$ cujas probabilidades de acesso são dadas por $p_1, \ldots, p_n$. Um código de prefixo com restrição de comprimento L é um código de prefixo cujo comprimento de todas as palavras que compõe o código são menores ou iguais a L, onde L é um inteiro tal que $L \geq \lceil \log n \rceil$.

Seja $\epsilon = \sum_{i=1}^{n} p_i.\bar{l}_i - \sum_{i=1}^{n} p_i.l_i$ a ineficiência do código de prefixo ótimo com restrição de comprimento L, onde $\bar{l}_1, \ldots, \bar{l}_n$ são os comprimentos das codewords do código ótimo com restrição de comprimento L e $l_1, \ldots, l_n$ são os comprimentos das codewords do código ótimo sem restrição de comprimento para o alfabeto Σ.

Seja ψ a razão áurea igual a 1,618. Neste artigo, mostramos que $\epsilon < 1/\psi^{L-\lceil \log(n+\lceil \log n \rceil - L) \rceil - 1}$ quando $L > \lceil \log n \rceil$. Através da exibição da cota inferior $\frac{1}{\psi^{L-\lceil \log n \rceil + 2 + \lceil \log \frac{n}{n-L} \rceil} - 1}$ para o máximo valor ϵ, mostramos que a cota superior obtida é assintoticamente ótima quando L está na faixa $\lceil \log n \rceil < L \leq n/2$. Para o caso $L = \lceil \log n \rceil$, provamos a cota $\epsilon < \lceil \log n \rceil - 1$ que é justa.

Além disto, apresentamos um algoritmo $1/\psi^{L-\lceil \log(n+\lceil \log n \rceil - L) \rceil - 1}$-aproximativo para construção de códigos de prefixo com restrição de comprimento L. Este algoritmo tem complexidade de espaço e tempo iguais a $O(n)$ se as probabilidades fornecidas estiverem ordenadas.

Os resultados apresentados indicam que os códigos de prefixo com restrição comprimento podem ser implementados eficientemente produzindo códigos cuja perda de compressão é bastante pequena.

Palavras-Chave: Códigos de Prefixo, Árvores de Huffman, Algoritmos Aproximativos

1 Introduction

An important problem in the field of Coding and Information Theory is the binary prefix code problem. Given an alphabet $\Sigma = \{a_1, \ldots, a_n\}$ and a corresponding set of strictly positive probabilities $\{p_1, \ldots, p_n\}$, with $\sum_{i=1}^{n} p_i = 1$, the problem is to find a prefix code for Σ, that minimizes the *average length* of a codeword, defined to be $\sum_{i=1}^{n} p_i l_i$, where l_i is the length of the codeword assigned to a_i. Such a code can be constructed by Huffman's Algorithm [Huf52, Ktm95]. The average length of a code gives the number of bits per symbol used by such a code.

The $L-$restricted prefix code problem consists in minimizing $\sum_{i=1}^{n} p_i l_i$ constrained to $l_i \leq L$ for $i = 1, \ldots, n$, where L is such that $L \geq \lceil \log n \rceil$. Gilbert [Gil71], recommends to formulate this problem when the weights w_i are inaccurately known. The L-restricted prefix codes can be useful in full-text retrieval applications, where the source alphabet is usually large [Tur95]. If there is no restriction on code lengths, the encoder can produce codewords that do not fit into a word, what can result in significant decoding speed degradation. Another possible use is on hardware implementation of such codes, where the space for storing the codes may be limited and each additional space may have a considerable cost.

Some methods can be found in the literature to solve the restricted problem [HuT72, Voo74, Gar74, Fra93]. Larmore and Hirschberg [Lar90], present an $O(nL)$ algorithm that requires linear space. Schieber [Sch95] presents an $O(n 2^{o(\sqrt{\log L \log\log n})})$ algorithm. Milidiú and Laber [Mil97a, Lab97] present an $O(n \log n + n \log \overline{w}_n)$ algorithm that constructs suboptimal restricted codes, where $\overline{w}_n$ is the highest given weight. This last algorithm allows a simple in-place implementation [Mil97b].

Given a set of probabilities $p_1, \ldots, p_n$ and a length L, we define the average code length difference ϵ as $\sum_{i=1}^{n} p_i.\overline{l_i} - \sum_{i=1}^{n} p_i.l_i$, where $\overline{l_1}, \ldots, \overline{l_n}$ are the lengths of the optimal $L-$restricted prefix code and $l_1, \ldots, l_n$ are the lengths of an optimal unrestricted prefix code. The value of ϵ cannot be smaller than 0, since the average length of the optimal unrestricted code cannot be greater than that of the optimal $L-$restricted code. Depending on the given weights and the length L, ϵ assumes different values. Some practical experiments have shown that the value of ϵ is small [Fra93, Mof97], even when the value of L is close to $\lceil \log n \rceil$. In this paper, we give some upper bounds on ϵ. As an immediate consequence of those bounds, we have that $\epsilon \leq 0.01$, whenever $L \geq \lceil \log n \rceil + 11$. This result assures that the loss of compression implied by the use of length restricted codes is almost negligible when $L \geq \lceil \log n \rceil + 11$.

Our upper bound is derived from the analysis of a new algorithm for constructing L-restricted prefix codes. This algorithm called BRCI is an approximative algorithm runs in $O(n \log n)$ time and requires $O(n)$ space. If the given probabilities are already sorted, the algorithm runs in $O(n)$ time. It must be noticed that all the exact algorithms have higher complexities.

The BRCI algorithm has some similarity with the ROT algorithm proposed by Fraenkel [Fra93]. Both of them construct an optimal code tree. Next, they construct a code tree with restricted maximal height L, by removing some leaves of the optimal code tree and condensing them into a complete binary tree. However, there is a fundamental difference: Fraenkel's algorithm does not have an approximation bound based on the input data L, n and $p_1, \ldots, p_n$; whereas for the BRCI it is proved an $1/\psi^{L-\lceil \log(n+\lceil \log n \rceil - L) \rceil - 1}$ approximation. Another difference, is that Fraenkel's construction is based on a cannonical optimal code tree, while ours is based on an ordinary Huffman tree.

The paper is organized as follows. In section 2, we present some concepts and results that are useful throughout this paper. In section 3, we present the BRCI algorithm. In section 4, we present some upper bounds on ϵ and a lower bound for the maximum value of ϵ. Finally, in section 5 we summarize our findings and conclusions.

We believe that the main contribution of this paper is to give some insight on when the use of L-restricted prefix codes provides a small loss of compression.

2 Prefix Codes and Binary Trees

2.1 Huffman Trees

The binary prefix code problem is equivalent to the following problem: given a set of probabilities $p_1, \ldots, p_n$, with $\sum_{i=1}^{n} p_i = 1$, find a binary tree T that minimizes the *weighted path length* $\sum_{i=1}^{n} p_i l_i$, where l_i is the level of the ith leaf of T. The level of a node is the number of edges between a node and the root of the tree. The tree T must be chosen among all binary trees with n leaves.

The equivalence is due to the fact that every prefix code can be represented by a binary tree [Eve79]. A tree that minimizes the external weighted path length for a given set of probabilities can be obtained by Huffman's algorithm. The trees obtained by Huffman's algorithm are usually called Huffman trees. Since Huffman's algorithm always construct a full binary tree [1], every Huffman tree is a full binary tree. An important relation concerning the levels of the leaves of a full binary tree is the following one

$$\sum_{i=1}^{n} 2^{-l_i} = 1 \qquad (1)$$

The $L-$restricted prefix code problem also admits a tree formulation, but with the additional restriction that the height of the tree cannot exceed L. Throughout this work, we use optimal code tree with restricted maximal height L to denote a tree that solves the $L-$restricted prefix code problem.

2.2 A Characterization for the Huffman Trees

Based on Gallager's sibling property [Gal78], Knuth [Knu85], has shown that a binary tree T with n leaves is a Huffman tree, if and only if it satisfies the following properties:

> (i) To the n leaves have been assigned the probabilities $p_1, \ldots, p_n$ is some order, and to each internal node has been assigned a probability equal to the sum of the probabilities of its two children;
>
> (ii) The $2n-1$ nodes (external and internal) can be arranged in a sequence $y_1, \ldots, y_{2n-1}$ such that if ϕ_j is the probability of the node y_j we have $\phi_1 \leq \ldots \leq \phi_{2n-1}$, and such that nodes y_{2j-1} and y_{2j} are siblings, for $j = 1, \ldots, n$.

It is easy to realize that condition (i) implies that the probability of any node y_i is equal to the sum of the probabilities of the leaves of the subtree rooted at y_i. In particular, the probability of the root of a Huffman Tree is equal to $\sum_{i=1}^{n} p_i = 1$.

2.3 A Relation Between the Level of a Node and its Probability

Katona and Nemetz [Kat76], show that inequality $p < 1/F_{i+1}$, holds for all Huffman trees, where p is the probability of a node that is at level i, and F_i is the ith Fibonacci

[1] In a full binary tree, every internal node has exactly two sons

number, defined by the recursion $F_{i+1} = F_i + F_{i-1}$, $F_1 = F_2 = 1$. Now, we derive a stronger relation that will be useful to prove our bound on ϵ.

Theorem 1 *Let T be a Huffman tree and let F_i be the ith Fibonacci number. Hence, for each level i, at most one node of T that is at level i has its probability p satisfying $p > 1/F_{i+2}$.*

Let $y_1, \ldots, y_{2n-1}$ be the nodes of T, and let ϕ_j be the probability associated to node y_j. We assume that $\phi_j \leq \phi_{j+1}$, for $j = 1, \ldots, 2n$. We also assume that y_{2j-1} and y_{2j}, for $j = 1, \ldots, n$, are siblings in tree T. These assumptions are correct since T is a Huffman tree.

If $i = 0$, the result holds, since the only node at level 0 is the root of T. Now, let $i > 1$ and let us assume that there are two nodes $y_{i'_0}$ and y_{i_0} in T at level i, with probability greater than $1/F_{i+2}$. We show that this assumption leads us to a contradiction.

Without loss of generality, we can assume that $i'_0 < i_0$. Now, we prove by induction on $k = 1, \ldots, i$ that the ancestor y_{i_k} of node y_{i_0}, that is at level $i-k$, has its probability ϕ_{i_k} greater than F_{k+2}/F_{i+2}.

For $k = 1$ the result is true, since the sibling of node y_{i_0} has also probability greater than $1/F_{i+2}$ and as a consequence the parent of y_{i_0} has probability greater than $1/F_{i+2} + 1/F_{i+2} = F_3/F_{i+2}$. Now, we assume that the result holds for $k \leq j$ and we prove that it still holds for $k = j+1$. Let y_{i_j} and $y_{i_{j-1}}$, be respectively, the ancestor of y_{i_0} arranged on level $i - j$ and the ancestor of y_{i_0} arranged on level $i - (j - 1)$. It follows by induction that $\phi_{i_j} > F_{j+2}/F_{i+2}$ and $\phi_{i_{j-1}} > F_{j+1}/F_{i+2}$. Moreover, the sibling of node y_{i_j} has probability greater than F_{j+1}/F_{i+2}, otherwise we could obtain a tree with weighted path length smaller than T, by exchanging the node $y_{i_{j-1}}$ with the sibling of node y_{i_j}. Hence, the parent of node y_{i_j} has probability greater than $F_{j+1}/F_{i+2} + F_{j+2}/F_{i+2} = F_{j+3}/F_{i+2}$, what establishes our result.

If we apply this last result for $k = i$, we obtain that the ancestor of y_{i_0} that is at level 0 has probability greater than $F_{i+2}/F_{i+2} = 1$, that is, the root of T has probability greater than 1. This contradiction is enough to establish our theorem. Therefore, at most one node of T that is at level i has probability greater than $1/F_{i+2}$.

An immediately corollary is given

Corollary 1 *If T is a Huffman tree. For each level i, at most one node of T that is at level i has probability greater than $1/\psi^i$, where ψ is the golden ratio $\frac{1+\sqrt{5}}{2} \approx 1.618$.*

It's a well known result [Knu68] that $\psi^i < F_{i+2}$, where F_{i+2} is the $(i+2)-$th Fibonacci number. Hence, it follows from theorem 1 that at most one node of T that is at level i has probability greater than $1/\psi^i$.

2.4 The Number of Leaves on Levels greater than L

In this section, we give an upper bound for the maximum number of leaves that can at levels greater than or equal L in a full binary tree. First, we present an useful lemma.

Lemma 1 *Let $k > 1$ be an integer number and let $a_1 \leq \ldots \leq a_k$ be an ordered sequence of positive integers. If $\sum_{i=1}^{k} 2^{-a_i} < 1$, then $\sum_{i=1}^{k} 2^{-a_i} \leq 1 - 2^{-k}$.*

We prove by induction on the value of k. First, we prove for $k = 2$. If $2^{-a_1} + 2^{-a_2} < 1$, then we must have $a_2 \geq 2$, otherwise $a_1 = a_2 = 1$ and $2^{-a_1} + 2^{-a_2} = 1$. Hence, $2^{-a_1} + 2^{-a_2} \leq 2^1 + 2^{-2} = 1 - 2^{-2}$.

Now, we assume that the result holds for $k = m$ and we prove that it also holds for $k = m + 1$. We divide the proof into two cases.

Case 1: $a_{m+1} > m + 1$. The induction hypothesis assures that $\sum_{i=1}^{m} 2^{-a_i} \leq 1 - 2^m$. Hence, $\sum_{i=1}^{m+1} 2^{-a_i} \leq 1 - 2^{-m} + 2^{-(m+2)} < 1 - 2^{-(m+1)}$.

Case 2: $a_{m+1} \leq m + 1$. Since 2^{-a_i}, for $i = 1, \ldots, m + 1$ is divisible by $2^{-a_{m+1}}$, it follows that $\sum_{i=1}^{m+1} 2^{-a_i}$ is an integral multiple of $2^{-a_{m+1}}$. Hence, we can rewrite $\sum_{i=1}^{m+1} 2^{-a_i} = I \times 2^{-a_{m+1}}$, where I is a positive integer. Since $a_{m+1} \leq m + 1$, we must have $I \leq 2^{m+1} - 1$, otherwise $I \times 2^{-a_{m+1}} \geq 1$. Therefore, we have $I \times 2^{-a_{m+1}} \leq (2^{m+1} - 1) \times 2^{-(m+1)} = 1 - 2^{-(m+1)}$.

Theorem 2 *Let T be a full binary tree with n leaves and height greater than L, where L is an integer number. Then, the number of leaves in T arranged on levels greater than or equal L is smaller than $n + \lceil \log n \rceil - L$.*

Let $0 < l_1 \leq \ldots \leq l_n$ be the levels of the leaves of T. Let M be number of leaves of T at levels greater than or equal L. Since T is a full binary tree and its height is greater than L, then M must be greater than 1.

It follows from the previous assumption that $L \leq l_{n-M+1} \leq \ldots \leq l_n$. Hence, we must have

$$\sum_{i=n-M+1}^{n} 2^{-l_i} < M \times 2^{-L} \leq 2^{\lceil \log M \rceil - L} \tag{2}$$

Since T is a full binary tree, it follows that $\sum_{i=1}^{n} 2^{-l_i} = 1$. Hence, $\sum_{i=1}^{n-M} 2^{-l_i} < 1$. Under this condition, lemma 1 assures that

$$\sum_{i=1}^{n-M} 2^{-l_i} \leq 1 - 2^{M-n} \tag{3}$$

By adding inequalities (2) and (3), we obtain

$$\sum_{i=1}^{n} 2^{-l_i} < 1 - 2^{M-n} + 2^{\lceil \log M \rceil - L} \tag{4}$$

Since $\sum_{i=1}^{n} 2^{-l_i} = 1$, it follows from equation (4) that $\lceil \log M \rceil - L > M - n$. Rearranging the terms, we obtain $M < n + \lceil \log M \rceil - L$ and hence $M < n + \lceil \log n \rceil - L$, because $M \leq n$.

Since every Huffman tree is a full binary tree, then the previous result holds for any Huffman tree.

3 The BRCI Algorithm

Let us consider a given ordered set of probabilities $0 < p_1 \leq \ldots \leq p_n$ and an integer L, with $L > \lceil \log n \rceil$. We use T to denote a Huffman tree for these probabilities.

In this section, we present an algorithm that obtains a code tree T_R, with height not greater than L, by performing some modifications on T's topology. The complexity of the algorithm and the average code length of the tree T_R are also discussed. This algorithm provides a constructive approach to the key bound on ϵ that we present in section 4.

vThe algorithm BRCI that constructs the tree T_R is divided into five steps as shown figure 1. The acronym BRCI comes from the initials of Build, Remove, Condense and Insert. These are the main operations performed by the algorithm. The word condense is due to the fact that the leaves removed in Step 1 are condensed into a complete binary tree in Step 2. The other words are used with obvious meaning.

In order to illustrate how the algorithm works, consider the set of probabilities $\{1/192, 2/192, 3/192, 4/192, 5/192, 6/192, 9/192, 18/192, 30/192, 41/192, 61/192\}$ and let $L = 5$. At step 0, a Huffman tree T for the given set of probabilities is constructed as shown in figure 2.(a). The height of T is equal to 7.

At Step 1, all subtrees rooted at nodes at level 5 are removed from T. At the end of this step, the tree of figure 2.(b) is obtained. At step 2, the leaves that were

BRCI Algorithm

Step 0:

Build a Huffman tree T for the given set of probabilities.

Step 1: Remove all subtrees of T that are rooted at nodes arranged on level L. Let T^1 be the remaining tree.

Step 2: Create a complete binary tree T^2 with the leaves of the subtrees removed at Step 1. Let $h(T^2)$ be the height of T^2.

Step 3: Let y_p be the node in T^1 with minimum probability arranged on level $L - h(T^2) - 1$ in T^1 and let T_p be the subtree rooted at y_p. Create a new node y^*. If y_p is not the root of T^1, then insert the node y^* between y_p and the parent of y_p. If y_p is the root of T^1, then insert T_p under y^*. Let T^3 be the remaining tree.

Step 4: Insert the tree T^2 under the node y^* in tree T^3. Let T_R be the obtained tree.

Figure 1: The BRCI Algorithm

removed from T in step 1, are used to construct a complete binary tree T^2 that is shown in figure 2.(c). Observe that $h(T^2) = 3$. At step 3, a node y^* is created and becomes the parent of the node with minimum probability at level $5 - 3 - 1 = 1$. Such a situation is illustrated in figure 2.(d). Finally, at step 4, the tree T^2 is inserted under the node y^* as shows figure 2.(e). The height of the tree T_R, obtained after step 4, is equal to 5. Furthermore, the average length of the code induced by such a tree is equal to 3.09375, while the average length of the code induced by T is 2.84375.

3.1 The Algorithm Analysis

In this subsection, we analyze the complexity and the approximation of the BRCI algorithm.

Let A and A_R be, respectively, the average length of the codes induced by the trees T and T_R. In the next theorem, we prove that the height of T_R is exactly L and that $A_R - A < 1/\psi^{L-\lceil\log(n+\lceil\log n\rceil-L)\rceil-1}$.

Theorem 3 *The tree T_R constructed by BRCI is a binary tree with height exactly L. Furthermore, we have that $A_R - A \leq 1/\psi^{L-\lceil\log(n+\lceil\log n\rceil-L)\rceil-1}$.*

The tree T^1 obtained after step 1 has height $L-1$, since the algorithm removes all the subtrees of T that are at level L. When the node $y*$ is inserted in the third step, the height of T_1 either increases by one unity or remains the same. In effect, if the subtree T_p has a leaf on level $L-1$ in T^1, then the height of T_1 increases by one unity. Otherwise, the height of T_1 remains the same. Hence, the height of T^3 is either $L-1$ or L. Finally, in the fourth step, the tree T^2 is inserted under the node y^* in tree T^3. Since the height of T^2 is equal $h(T^2)$ and the node y^* is at level $L-h(T^2)-1$, then this insertion produces a tree with height exactly L.

To complete our demonstration, we must prove the result concerning the average code length. By comparing T and T_R, we observe that the levels of the leaves of T_p have increased by one unity in T_R. This increment happens at step 3 of BRCI. On the other hand, the levels of the other leaves of T, either decrease or remain the same in tree T_R. Hence, we have

$$A_R \leq A + S_p,$$

where S_p is the sum of the leaf probabilities of T_p.

At this point, recall that the sum of the leaf probabilities of a subtree in a Huffman tree is equal to the probability of the subtree's root. Since the leaves at levels not smaller than L were removed from T in step 1, then the value of S_p is not greater than the probability of the node with minimum probability at level $L-h(T^2)-1$ in the tree T. Hence, it follows by corollary 1 that $S_p \leq 1/\psi^{L-h(T^2)-1}$. On the other hand, theorem 2 assures that the number of the leaves of T^2 is not greater than $n+\lceil\log n\rceil - L$. Since T^2 is a complete binary tree, then we must have $h(T^2) \leq \lceil\log(n+\lceil\log n\rceil - L)\rceil$. Therefore, we obtain that $S_p \leq 1/\psi^{L-\lceil\log(n+\lceil\log n\rceil-L)\rceil-1}$ and as a consequence $A_R - A \leq 1/\psi^{L-\lceil\log(n+\lceil\log n\rceil-L)\rceil-1}$.

Now, let T_L be an optimal code tree with restricted maximal length L for the given set of probabilities $p_1,\ldots,p_n$. Furthermore, let A_L be the average length of the code induced by the tree T_L. The next theorem proves the approximation and the complexity of the BRCI algorithm.

Theorem 4 *Given an integer $L > \lceil\log n\rceil$, the BRCI algorithm is a $1/\psi^{L-\lceil\log(n+\lceil\log n\rceil-L)\rceil-1}$-approximative algorithm for the construction of $L-$restricted prefix codes. Furthermore, the algorithm runs in $O(n)$ time and requires $O(n)$ space.*

In order to establish the approximation, we must prove that

$$\frac{A_R - A_L}{A_L} < 1/\psi^{L-\lceil\log(n+\lceil\log n\rceil-L)\rceil-1} \tag{5}$$

Since, $A < A_L$, it follows that $A_R - A_L < A_R - A$. On the other theorem 3 shows that $A_R - A \leq 1/\psi^{L-\lceil\log(n+\lceil\log n\rceil - L)\rceil - 1}$. Hence, we have $A_R - A_L < 1/\psi^{L-\lceil\log(n+\lceil\log n\rceil - L)\rceil - 1}$. Moreover, we have $A_L > 1$. That is true because the lengths of an optimal $L-$restricted prefix code must be greater than or equal to 1 and $\sum_{i=1}^{n} p_i = 1$. Therefore, we can conclude that inequality (5) holds, what assures the approximation.

Now, we must establish the complexity of the algorithm. If the probabilities are already sorted, the construction of the Huffman tree T performed in Step 0 can be done in $O(n)$ [Lee76, Ktm95]. The steps 1,3 and 4 can be performed in $O(n)$, since the algorithm just needs to search the tree, performing some insertions and deletions. The construction of a complete binary tree in step 2 also spends $O(n)$ time . Hence, if the given probabilities are already sorted, then BRCI constructs the tree T_R in $O(n)$ time.

The space consumed is just the space required for storing the trees T, T^1,T^3 and T_R and the space required to store the complete binary tree in step 2. Therefore, BRCI is an $O(n)$ time and space $1/\psi^{L-\lceil\log(n+\lceil\log n\rceil - L)\rceil - 1}$-approximative algorithm for the construction of $L-$restricted prefix codes.

4 Bounds on ϵ

Given a set of probabilities $p_1, \ldots, p_n$ and a length L, we define the average code length difference ϵ as $\sum_{i=1}^{n} p_i.\overline{l_i} - \sum_{i=1}^{n} p_i.l_i$, where $\overline{l_1}, \ldots, \overline{l_n}$ are the lengths of the optimal $L-$restricted prefix code and $l_1, \ldots, l_n$ are the lengths of an optimal unrestricted prefix code.

In this section we present some bounds on ϵ. First, we present some upper bounds on ϵ and after we present a lower bound for the maximum value of ϵ.

4.1 Upper Bounds

Let T be a Huffman tree for the given set of probabilities $0 < p_1 \leq \ldots \leq p_n$, and let L be an integer, with $L \geq \lceil\log n\rceil$. We use $l_1, \ldots, l_n$ to denote the levels of the leaves of T. If L is greater than or equal to the height of T, we have that $\epsilon = 0$. Therefore, we just analyze the case where L is smaller than the height of T.

We consider the cases $L = \lceil\log n\rceil$ and $L > \lceil\log n\rceil$ separately. First, let us examine the case where $L = \lceil\log n\rceil$.

Theorem 5 *If $L = \lceil\log n\rceil$, then $\epsilon < \lceil\log n\rceil - 1$. Furthermore, this bound is sharp.*

The bound is based on the following inequalities

$$\epsilon \leq \lceil \log n \rceil - \sum_{i=1}^{n} p_i l_i \tag{6}$$

$$\sum_{i=1}^{n} p_i l_i > 1 \tag{7}$$

Inequality (6) is true because the average length of the optimal $\lceil \log n \rceil$-restricted code is not greater than $\sum_{i=1}^{n} \lceil \log n \rceil p_i = \lceil \log n \rceil$. On the other hand, inequality (7) holds since $l_i \geq 1$ for $i = 1, \ldots, n$, $l_k \geq 2$ for some k, and $\sum_{i=1}^{n} p_i = 1$.

The sharpness of the bound can be assured, by considering the parametric set of probabilities given by $p_1 = \ldots = p_{n-1} = \delta$ and $p_n = 1 - (n-1) \times \delta$, where δ is an auxiliary parameter. If we assume that n is an integer power of 2, then the average length of the optimal $\lceil \log n \rceil -$ restricted code for these probabilities is equal to $\lceil \log n \rceil$. Moreover, if δ is small enough, then average length of the optimal unrestricted code is equal to $\sum_{i=1}^{n} p_i l_i = (1-(n-1)\delta) + \lceil \log n \rceil \delta + (n-1)(\lceil \log n \rceil + 1)\delta = 1 + n \lceil \log n \rceil \delta$. Hence, if δ goes to 0, then $\sum_{i=1}^{n} p_i l_i$ goes to 1 and $\epsilon \to \lceil \log n \rceil - 1$.

Figure 3 illustrates how to obtain the largest value for ϵ, when $L = \lceil \log n \rceil$. Figure 3.(a) shows a Huffman Tree for the weights $\delta, \delta, \delta, \delta, \delta, \delta, \delta, \delta, 1 - 7\delta$, and figure 3.(b) shows an optimal code tree with restricted maximal length 3 for the same weights.

Now, we consider the case where $L > \lceil \log n \rceil$. The bound for this case is an immediate consequence of theorem 3.

Theorem 6 *If $L > \lceil \log n \rceil$, then $\epsilon \leq 1/\psi^{L - \lceil \log(n + \lceil \log n \rceil - L) \rceil - 1}$.*

Let A_L be the average length of an optimal $L-$restricted prefix code for the given probabilities $p_1, \ldots, p_n$. We have that, $A_L - A \leq A_R - A$, where A and A_R are, respectively, the average length of the Huffman code and the code generated by the BRCI algorithm for the same set of probabilities. Since theorem 3 assures that $A_R - A \leq 1/\psi^{L - \lceil \log(n + \lceil \log n \rceil - L) \rceil - 1}$, it follows that $A_L - A = \epsilon \leq 1/\psi^{L - \lceil \log(n + \lceil \log n \rceil - L) \rceil - 1}$.

When some data is encoded with optimal $L-$restricted codes, the achieved bound implies that the number of additional bits required per symbol decreases exponentially with L. Of course, L must be greater than $\lceil \log n \rceil$. For example, if $L = \lceil \log n \rceil + 11$, then the average number of additional bits is smaller than 0.01. If we set $L = \lceil \log n \rceil + 15$, then the average number of additional bits is no greater than 0.001.

4.2 Lower Bounds

In this section, we present a lower bound for the maximum value of ϵ, by introducing a generic example, where $\epsilon \geq \frac{1}{\psi^{L-\lceil \log n \rceil + 2 + \lceil \log \frac{n}{n-L} \rceil} - 1}$.

In order to simplify the notation, we work with symbol weights instead of symbol probabilities. If the set of weights is given by $w_1, \ldots, w_n$, then the set of probabilities is given by $p_1, \ldots, p_n$, with $p_i = w_i / \sum_{i=1}^{n} w_i$. Hence, whenever we present an example where the value of $\sum_{i=1}^{n} w_i l_i - w_i \bar{l}_i$ is greater than some value M, then the value of ϵ will be greater than $M / \sum_{i=1}^{n} w_i$.

Let k be an integer number, with $1 \leq k < n$ and let us construct the set of weights $W_k = \{w_1, \ldots, w_n\}$ that will be used on what follows. We divide these weights into two subsets. The subset of the "small" weights $w_1, \ldots, w_{n-k}$ and the subset of the "big" weights $w_{n-k+1}, \ldots, w_n$. All the small weights are equal to 1. On the other hand, the value of the big weights w_{n-k+i}, for $i = 1, \ldots, k$, are given by $x.F_i$, where x is a real number and F_i is the ith Fibonacci number. It is well known that $\sum_{i=1}^{k} F_i = F_{k+2} - 1$ [Knu68]. Hence, one can derive the following relation concerning the sum of the weights of W_k

$$\sum_{i=1}^{n} w_i = x(F_{k+2} - 1) + n - k \tag{8}$$

This last relation is useful for our purposes. In the next lemma, we construct a Huffman tree for the set W_k.

Lemma 2 *If $x > n - k$, then the Huffman tree for the set of weights W_k has height $k + \lceil \log(n-k) \rceil$.*

Let us consider a tree T with the following properties:

(a) For $i = 1, \ldots, k-1$, there is exactly one leaf and one internal node at level i of T. The leaf has weight $x.F_{k+1-i}$ and the internal node has weight $x.F_{k+2-i} - t$, where t is equal to $x - (n-k)$. Furthermore, the labels $2(n-i)-1$ and $2(n-i)$ were assigned, respectively, to the leaf and to the internal node at level i.

(b) There is exactly one leaf and one internal node at level k. The leaf has weight $x.F_1$ and the internal node has weight $n-k$. Furthermore, the internal node is the root of a Huffman tree T_s for the subset of the "small" weights. The labels $2(n-k)-1$ and $2(n-k)$ were assigned, respectively, to the internal node and to the leaf.

The tree T is illustrated in figure 4. The big gray triangle in the bottom of the tree represents a Huffman tree for the "small" weights. The weight of each node was written inside it, and the label was written at its left side. Now, we prove that T is a Huffman tree for the set of weights W_k, by showing that the nodes of T satisfy conditions (i) and (ii) presented in section 2.2. Since, the nodes of T_s satisfy those conditions, we need only to check the nodes at levels not greater than k.

First, we must prove that an internal node's weight is equal to the sum of its two children weights. For the internal nodes at levels smaller than $k-1$, the result holds, since $xF_{k+2-i}-t = xF_{k+1-i}-t+xF_{k-i}$, for $i=0,\ldots,k-2$. For the internal node on level $k-1$ the result also holds, since $xF_3-t = x+(x-t) = xF_1+(n-k)$. Hence, condition (i) is valid.

The nodes $y_{2(n-i)}$ and $y_{2(n-i)-1}$ are siblings, since they are the unique nodes at level i. In order to conclude the prove, it must be shown that $\phi_j \leq \phi_{j+1}$, for $j \geq 2(n-k)-1$. This fact can be easily verified, by considering the case where the nodes y_j and y_{j+1} are at the same level, and the case where these nodes are at consecutive levels.

We have proved that T is a Huffman tree for the set of weights W_k. Since the root of the Huffman tree for the "small weights" is on level k and the height of a Huffman tree for a set of $n-k$ equal weights is equal $\lceil\log(n-k)\rceil$, then the height of T is given by $k+\lceil\log(n-k)\rceil$.

Before we present a lower bound for the maximum value of ϵ, we need one more lemma.

Lemma 3 *Let k and L be integer numbers, with $1 \leq k < n$ and $\lceil\log n\rceil \leq L \leq n-2$. Then, the minimum value of k that satisfies $k+\lceil\log(n-k)\rceil-1 = L$ is such that*

$$k \leq L - \lceil\log n\rceil + 1 + \lceil\log(\frac{n}{n-L})\rceil.$$

Since $k+\lceil\log(n-k)\rceil-1$ is a non decreasing function of k, then we just need to prove that that $k+\lceil\log(n-k)\rceil-1 \geq L$, when $k = L-\lceil\log n\rceil+1+\lceil\log(\frac{n}{n-L})\rceil$. Letting $\alpha = \frac{n}{n-L}$, we must prove that

$$L - \lceil\log n\rceil + 1 + \lceil\log\alpha\rceil + \lceil\log(n/\alpha + \lceil\log n\rceil - 1 - \lceil\log\alpha\rceil)\rceil - 1 \geq L$$

That is,

$$\lceil\log(n/\alpha + \lceil\log n\rceil - 1 - \lceil\log\alpha\rceil)\rceil - \lceil\log n\rceil + \lceil\log\alpha\rceil \geq 0 \tag{9}$$

Since $L \leq n-2$, then $\lceil\log\alpha\rceil \leq \lceil\log n\rceil - 1$. Hence, inequality (9) holds if

$$\lceil\log(n/\alpha)\rceil - \lceil\log n\rceil + \lceil\log\alpha\rceil \geq 0. \tag{10}$$

We can rewrite inequality (10) as

$$\lceil \log n - \log \alpha \rceil \geq \lceil \log n \rceil - \lceil \log \alpha \rceil \tag{11}$$

However, inequality (11) is a well known property of the ceiling function. Therefore, the lemma holds.

The next theorem gives a lower bound for the maximum value of ϵ.

Theorem 7 *Given the integers n and $L > \lceil \log n \rceil$, it is always possible to find a set of weights $w_1, \ldots, w_n$ such that the value of ϵ is not smaller than $\frac{1}{\psi^{L - \lceil \log n \rceil + 2 + \lceil \log \frac{n}{n-L} \rceil} - 1}$.*

Let k be the smallest integer such that $L = k + \lceil \log(n-k) \rceil - 1$ and let T and T_L be, respectively, a Huffman tree and an optimal code tree with restricted maximal length L for the set of weights W_k already defined in this section. Let also $l_1, \ldots, l_n$ and $\bar{l}_1, \ldots, \bar{l}_n$ be, respectively, the levels of the leaves of T and T_L.

The external weighted path length of T and T_L are, respectively, given by

$$\sum_{i=1}^{n} w_i l_i = \sum_{i=1}^{n-k} l_i + \sum_{i=n-k+1}^{n} x F_{i-n-k} l_i \tag{12}$$

and

$$\sum_{i=1}^{n} w_i \bar{l}_i = \sum_{i=1}^{n-k} \bar{l}_i + \sum_{i=n-k+1}^{n} x F_{i-n-k} \bar{l}_i \tag{13}$$

Since the height of T_L is not greater than $k + \lceil \log(n-k) \rceil - 1$, then $\bar{l}_i \leq k + \lceil \log(n-k) \rceil - 1$, for $i = 1, \ldots, n$. On the other hand, $l_1, \ldots, l_{n-k}$ can assume only the values $k + \lceil \log(n-k) \rceil$ or $k + \lceil \log(n-k) \rceil - 1$. This happens, since the height of T is equal to $k + \lceil \log(n-k) \rceil$, and equal weights must be at consecutive levels in a Huffman Tree [Mil97a]. Hence, we have

$$\sum_{i=1}^{n-k} \bar{l}_i < \sum_{i=1}^{n-k} l_i. \tag{14}$$

The optimality of T assures that $\sum_{i=1}^{n} w_i l_i \leq \sum_{i=1}^{n} w_i \bar{l}_i$. Therefore, it follows from equations (12), (13) and (14) that

$$\sum_{i=n-k+1}^{n} x F_{i-n-k} l_i < \sum_{i=n-k+1}^{n} x F_{i-n-k} \bar{l}_i \tag{15}$$

We can observe that the two members of inequality (15) are integer multiples of x. Hence, we have

$$\sum_{i=n-k+1}^{n} xF_{i-n-k}\bar{l}_i - \sum_{i=n-k+1}^{n} xF_{i-n-k}l_i \geq x \tag{16}$$

Since $l_i \leq k + \lceil \log(n-k) \rceil$ and $\bar{l}_i > 0$, then we have the following inequality

$$\sum_{i=1}^{n-k} \bar{l}_i - \sum_{i=1}^{n-k} l_i \geq -(n-k)(k + \lceil \log(n-k) \rceil) \tag{17}$$

Adding inequalities (16) and (17), we obtain that

$$\sum_{i=1}^{n} w_i\bar{l}_i - \sum_{i=1}^{n} w_i l_i \geq x - (n-k)(k + \lceil \log(n-k) \rceil) \tag{18}$$

Since $\sum_{i=1}^{n} w_i = x(F_{k+2} - 1) + (n-k)$, then it follows that

$$\epsilon = \frac{\sum_{i=1}^{n} w_i\bar{l}_i - \sum_{i=1}^{n} w_i l_i}{\sum_{i=1}^{n} w_i} \geq \frac{x - (n-k)(k + \lceil \log(n-k) \rceil)}{x(F_{k+2} - 1) + (n-k)} \tag{19}$$

Hence, by increasing the value of x we can let ϵ be arbitrarily close to $1/(F_{k+2} - 1)$. Since $k + \lceil \log(n-k) \rceil - 1 = L$, it follows from lemma 3 that $k \leq L - \lceil \log n \rceil + 1 + \lceil \log \frac{n}{n-L} \rceil$. Hence, for the set of weights W_k, we have

$$\epsilon \geq \frac{1}{\psi^{L - \lceil \log n \rceil + 2 + \lceil \log \frac{n}{n-L} \rceil} - 1},$$

what establishes the theorem.

The lower bound presented in the previous theorem does not assure that the upper bound proved in theorem 6 is tight. However, in the most usual range, that is, for $\lceil \log n \rceil < L \leq n/2$, the upper bound is asymptotically tight. That is true because the value of the upper bound in this range is equal to $1/\psi^{L - \lceil \log n \rceil - 1}$ and the value of the lower bound is equal $1/\psi^{L - \lceil \log n \rceil + 2}$. Hence, the two bounds differ approximately by a factor equal to $1/\psi^3$. Actually, by observing the bounds in terms of Fibonacci numbers, this factor can be improved to approximately $1/\psi^2$. The range $\lceil \log n \rceil < L \leq n/2$ is the most usual one, since for practical cases the Huffman tree height is often $\Theta(\log n)$.

Table 1 shows the values of ϵ achieved by compressing some text files that belong to the "Calgary Corpus". This corpus is a collection of 14 texts available [2] from the University of Calgary, Canada. For each file, we have determined an alphabet and its corresponding symbol weights, through an initial parsing. For some files, we use a word model to perform the parsing and for others we use a byte model.

[2]The Calgary Corpus is available by anonymous file-transfer protocol from ftp.cpsc.ucalgary.ca

Table 1: Values of ϵ obtained by compressing text files of the calgary corpus collection

Files	$L - \lceil \log n \rceil$								
	0	1	2	3	4	5	6	7	8
bib (81,16)	0.2988	0.0858	0.0288	0.0102	0.003	0.0011	0.0003	0.00009	0.00004
book1 (13186,18)	1.4738	0.3071	0.0976	0.0246					
book2 (7944,18)	3.0516	0.4372	0.1032	0.0234	0.0025				
geo (256,12)	2.3316	0.1388	0.0115	0.0009					
news(9975,17)	0.6438	0.1299	0.0219						
obj1(256,13)	2.0286	0.1582	0.0270	0.0307	0.0002				
obj2 (256,15)	1.7088	0.1798	0.0473	0.0139	0.0034	0.0006	0.0001		
paper1 (1834,14)	1.6001	0.2464	0.0345						
paper2 (2499,15)	0.5383	0.0987	0.0254						
pic (159,17)	0.9463	0.0902	0.0305	0.0118	0.0046	0.0015	0.0005	0.00012	0.00002
progc (1148,14)	0.5474	0.1106	0.0181						
progl (979,15)	2.7853	0.4780	0.1204	0.0327	0.0056				
progp (595,15)	0.6166	0.1775	0.0715	0.0153	0.0024				
trans (99,16)	0.4841	0.1381	0.0438	0.0152	0.0049	0.0014	0.0004	0.00009	0.00001
Bound Proved in Theorem 6	******	1.0000	0.5000	0.3333	0.2000	0.1250	0.0769	0.04761	0.02941

Next, we have computed the optimal unrestricted prefix code and the optimal $L-$restricted prefix code for each file. The restricted code, was computed for L running from $\lceil \log n \rceil$ to $H-1$, where H is the length of the longest Huffman code for the file. The value of n is given by the first number in parenthesis on each table entry, and the value of H is given by the second number. The blank values in the table correspond to length restrictions greater than or equal to H. In these ranges, the value of ϵ is 0. We have also computed the value of the bound proved in theorem 6. The corresponding values are listed in the last row of table 1.

The bound proved in theorem 6 shows that $\epsilon \leq 0.01$ whenever $L \geq \lceil \log n \rceil + 11$. Hence, the loss of compression implied by the usage of length restricted codes is almost negligible when $L \geq \lceil \log n \rceil + 11$. On the other hand, it must be observed that none of the values shown in Table 1 are close to that bound. These results suggest that the average code length difference is almost negligible for practical purposes when $L \geq \lceil \log n \rceil + 4$.

5 Conclusions

Throughout this work, we have established an upper bound concerning the average code length difference implied by the usage of length restricted prefix codes. We have also presented an $1/\psi^{L-\lceil\log(n+\lceil\log n\rceil-L)\rceil-1}$-approximate algorithm for the $L-$restricted prefix code problem. If the given alphabet probabilities are already sorted, the algorithm has $O(n)$ time complexity, otherwise the complexity is $O(n\log n)$. The space required is always $O(n)$.

The bound proved in theorem 6 shows that if L is a little greater than $\lceil\log n\rceil$, then the average number of bits required to encode an alphabet using $L-$restricted prefix codes is very close to that required when using unrestricted prefix codes. Hence, if the value of L is carefully chosen, one can safely utilize $L-$restricted prefix codes without being concerned about compression degradation.

The results presented in this paper indicate that one can efficiently implement prefix codes with length restriction, obtaining also very effective codes.

Acknowledgements

To Artur Alves Pessoa for the Lower Bound example and for some important comments and discussions.

References

[Eve79] Even, S. *Graph Algorithms*, Computer Science Press, 1979.

[Fra93] Fraenkel, A.S., and Klein, S. T. *Bounding the depth of search trees*, The Computer Journal, 36, (7), 668-678 (1993).

[Gal78] Gallager, R. G., *Variations on a theme by Huffman*, IEEE Transactions on Information Theory., IT-24, 668-674, (1978).

[Gar74] Garey, M. R. *Optimal binary search trees with restricted maximal depth* SIAM J. Computing 23(1974), 101-110

[Gil71] Gilbert, E.N., *Codes based on innacurate source probabilities* IEEE Transactions on Information Theory., IT-17, 304-314, (1971).

[HuT72] Hu, T. C., and Tan, K. C., *Path length of binary search trees.*, SIAM J. Appl Math 22 (1972), 225-234

[Kat76] Katona, G. O. H., and Nemetz O. H. T., *Huffman codes and self information* IEEE Transactions on Information Theory., IT-22, 337-340, (1976).

[Knu68] Knuth, D. E. *The Art of Computer Programming 1: Fundamental Algorithms*, Addison Wesley, Reading, Mass., 1968.

[Knu85] Knuth,D.E.*Dynamic Huffman Coding*, Journal of Algorithms 6,2 (1985), 163-180.

[Ktm95] Katajainen, J., Moffat, A., *Inplace calculation of minimum-redundancy codes*, In Proc. Workshop an Algorithms and Data Structures, Kingston University, Canada, August 1995. Springer-Verlag.

[Huf52] Huffman, D.A. *A Method for the Construction of the Minimum Redundancy Codes*, Proc. Inst. Radio Engineers, Vol 40,No 10, 1098-1101, September 1952.

[Lab97] Laber, E. S., Um Algoritmo Eficiente para Construção de Códigos de Huffman com Restrição de Comprimento, M.Sc. Dissertation, Departamento de Informática, PUC-RJ, Brasil, 1997.

[Lar90] Larmore, L.L. and Hirschberg, D.S. *A Fast Algorithm for Optimal Lenght-Limited Huffman Codes*, Journal of the ACM, July (1990), 464-473.

[Lee76] Leeuwen, J. van (1976) On the construction of Huffman trees. In proc. 3rd Int. Coll. On Automata, Languages, and Programming, pp 382-410

[Mil97a] Milidiú, R.L. and Laber, E.S., *The WARM-UP ALGORITHM: A Lagrangean Construction of Length Restricted Huffman Codes*, Submitted to SIAM Journal on Computing.

[Mil97b] Milidiú, R.L., Pessoa A. A. and Laber, E.S., *Practical use of the WARM-UP Algorithm on Length-Restricted Codes*, To appear in the Proceedings of the Fourth Latin American Workshop on String Processing, Chile, 1997.

[Mof97] Moffat A. and Turpin A. *Efficient Construction of Minimum-Redundancy Codes For Large Alphabets* To appear in IEEE Transactions on Information Theory.

[Sch95] Schieber B. *Computing a minimum-weight k-link path in graphs with the concave monge property.* In Proc. 6th ACM IEEE Symposium on Discrete Algorithms, pages 405-411,San Francisco, January 1995.

[Sch64] Schwartz, E. S. *An Optimum Encoding with Minimum Longest Code and Total Number of Digits*, Information and Control 7, 37-44 (1964).

[Tur95] Turpin A. and Moffat A. *Practical Length-limited Code for Large Alphabets*, The Computer Journal, 38, (5), 339-347 (1995).

[Voo74] Van Voorish D. C., *Constructing codes with bounded codeword lengths*, IEEE Transactions on Information Theory., IT-20, 288-290, (1974).

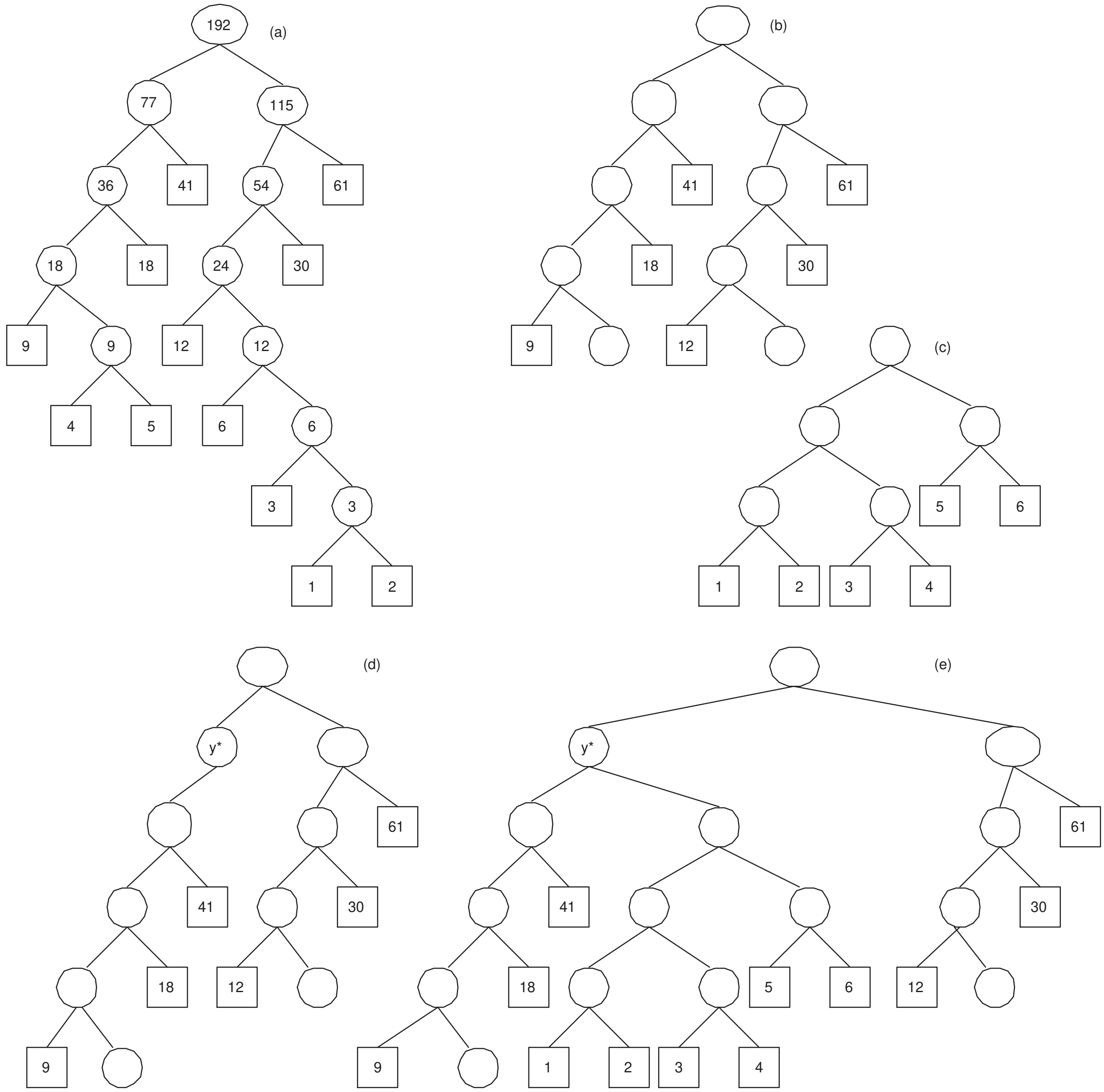

Figure 2: The trees constructed during the execution of the algorithm BRCI. Figure (a) shows a Huffman tree for the probabilities $\{1/192, 2/192, 3/192, 4/192, 5/192, 6/192, 9/192, 18/192, 30/192, 41/192, 61/192\}$. Figure (b) shows the tree T^1 obtained after the deletions of step 1. Figure (c) shows the complete binary tree constructed in step 2. Figure (d) shows the tree obtained after the inclusion of node y^*. Figure (e) shows the tree T_R returned by the BRCI algorithm.

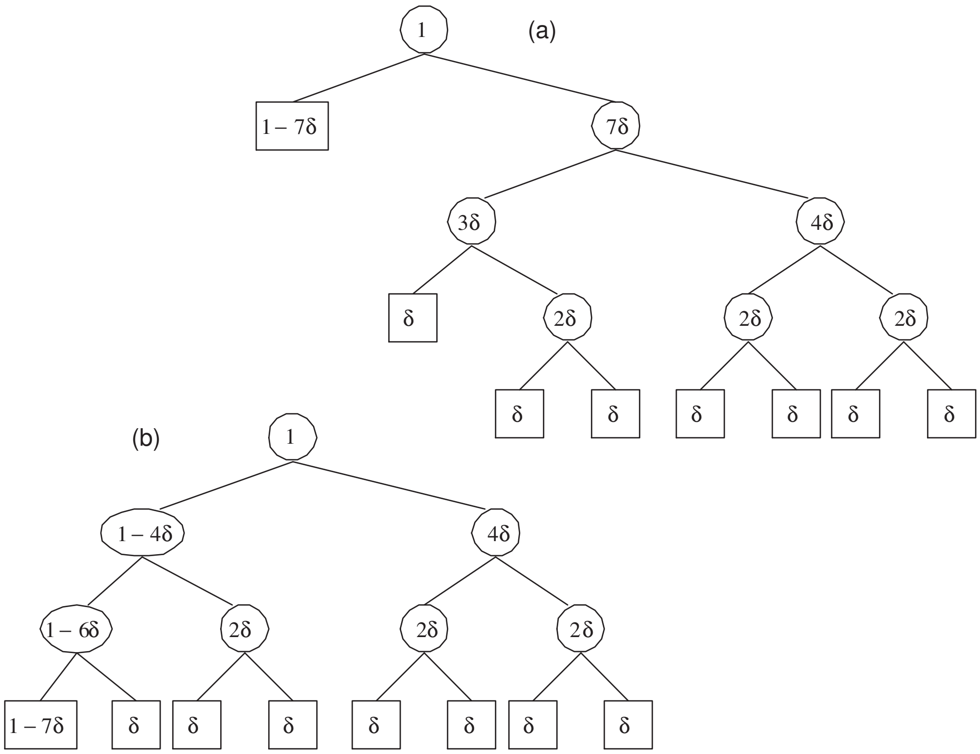

Figure 3: Generating the maximum value for ϵ, when $L = \lceil \log n \rceil$

.

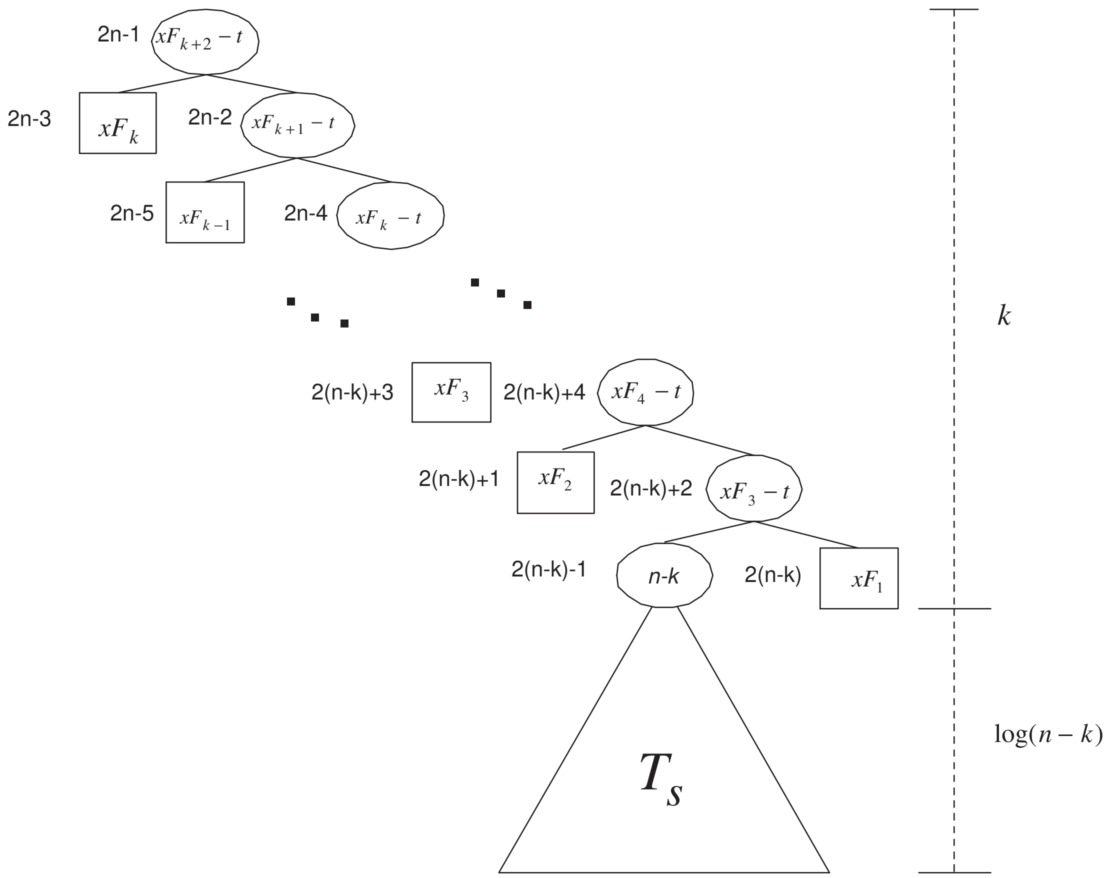

Figure 4: A Huffman tree for the set of weights W_k. The value of x is such that $x > n - k$.