Technical Report CUCS-031-93

New Lower Bounds on the Cost of Binary Search Trees*

Roberto De Prisco[†] **Alfredo De Santis**[‡]

Abstract

In this paper we provide new lower bounds on the cost of binary search trees. The bounds are expressed in terms of the entropy of the probability distribution, the number of elements and the probability that a search is successfully. Most of our lower bounds are derived by means of a new technique which exploits the relation between trees and codes. Our lower bounds compare favorably with known limitations.

We also provide an achievable upper bound on the Kraft sum generalized to the internal nodes of a tree. This improves on a previous result.

*This work was partially supported by the National Council of Research (C.N.R.) under grant 91.02326.CT12 and by M.U.R.S.T. in the framework of Project: "Algoritmi, Sistemi di Calcolo e Strutture Informative".
[†]Department of Computer Science, Columbia University, New York, N.Y. 10027
[‡]Dipartimento di Informatica ed Applicazioni, Universitá di Salerno, 84081 Baronissi (SA) - Italy

1 Introduction

Binary search trees are a widely used data structure for information storage and retrieval. We are given n keys $K_1 < K_2 < ... < K_n$. When we want to search for a given key X there are exactly $2n+1$ possibilities, namely either X can be one of the keys K_i, for $i = 1, ..., n$ or X can be between K_i and K_{i+1}, for $i = 0, 1, ..., n$ (we assume that $K_0 = -\infty$ and $K_{n+1} = +\infty$, with obvious interpretation). We are also given a probability distribution $D = (q_1, ..., q_n; p_0, p_1, ..., p_n)$, over these $2n+1$ results due to a search for a particular key X. The probability that the searched key is K_i is q_i, whereas p_i is the probability that the searched key lies between K_i and K_{i+1}. Further we let $Q = \sum_{i=1}^{n} q_i$ and $P = \sum_{i=0}^{n} p_i$.

A binary search tree T is a tree with n internal nodes, that contain the keys K_i, and $n+1$ external nodes, that contain the intervals $]K_i, K_{i+1}[$, such that an inorder visit of the tree gives the keys and the intervals in the correct order.

We assign to each node a label: to the node that contains K_i we assign the label q_i, to the node that contains $]K_i, K_{i+1}[$ we assign the label p_i. We will use the label of a node as the name of the node. The level of q_i, denoted by $l(q_i)$, is the number of nodes from the root of T to q_i, whereas the level of p_i, denoted by $l(p_i)$, is the level of the parent of p_i.

If we are searching for a key X the level of q_i is the number of comparisons needed to retrieve X if $X = K_i$ and the level of p_i is the number of comparisons needed to establish that X lies between K_i and K_{i+1}. Hence we define the *cost* of the tree T as

$$C = \sum_{k=0}^{n} p_k l(p_k) + \sum_{k=1}^{n} q_k l(q_k).$$

An optimal binary search tree is a binary search tree that minimizes the cost C. We denote the cost of an optimal binary search tree by C_{opt}. It is clear that any lower bound for C_{opt} is a lower bound for the cost of any binary search tree. Hence, throughout this paper we consider only optimal binary search trees.

The entropy of the probability distribution D is[1]

$$H = \sum_{k=0}^{n} p_k \log \frac{1}{p_k} + \sum_{k=1}^{n} q_k \log \frac{1}{q_k}.$$

Mehlhorn [8] proved that

$$C_{opt} \geq H/\log 3 \tag{1}$$

and the smaller is H the tight is the bound. The above bound is expressed in terms of only the entropy of D. If other information on the probability distribution D are available, a better bound is known. The following lower bound [1],[2], holds:

$$C_{opt} \geq H - \log e - Q(\log\log(n+1) - 1) \tag{2}$$

and this bound improves on (1) for not small values of H (i.e., $H > 3.909 + 2.710Q \log\log(n+1) - 2.710Q$).

In this paper we introduce a technique which enables us to derive lower bounds on the cost of binary search trees starting from lower bounds on the expected codeword length of some

[1] Throughout this paper all logarithms are to base 2.

classes of codes. Exploiting this technique we provide lower bounds on C_{opt} which involve the knowledge of the entropy H of the probability distribution D, the number n of keys, and the probability Q that a search is successfully.

We derive three lower bounds which are function of H, n and Q that improve on (2) and two lower bounds which are function of H and Q only.

We also provide the following bound

$$C_{opt} \geq H - 1 - 2\log(H+2)$$

that improves on (1) for $H \geq \chi$, where $\chi \simeq 29.741$.

Finally, in deriving our bounds, we obtain an achievable upper bound on the Kraft sum generalized to the internal nodes of a tree that improves on a previous result.

The paper is organized as follows. In Section 2 we recall some useful notions and results. In Section 3 we derive the bounds and in Section 4 we obtain further improvements of the bounds derived in Section 3.

2 Preliminaries

In this section we recall some useful results that we will use in the rest of the paper. In deriving our bounds we exploit the relation between trees and codes to utilize some known lower bounds on the average codeword length of some classes of codes. In the following we briefly recall some notions about codes.

Let S be a source consisting of a set $\{a_1, a_2, ..., a_m\}$ of m letters and a probability distribution $(s_1, s_2, ..., s_m)$, where s_k denote the probability of letter a_k, $1 \leq k \leq m$. The entropy of the source S is the entropy of the probability distribution $(s_1, s_2, ..., s_m)$. A (binary) codeword is a sequence of bits. A code for S is a set of m codewords. Let $\mathcal{C} = \{x_1, x_2, \ldots, x_m\}$ be a code for source S and let $l(x_1), l(x_2), ..., l(x_m)$ be the codeword lengths. Codeword x_i encodes the letter a_i, for $i = 1, 2, \ldots, m$. The average codeword length of $\mathcal{C}$ is $L = \sum_{i=1}^{m} s_i l(x_i)$. We are interested in two particular classes of codes: prefix codes and one-to-one codes. A prefix code is a code in which no codeword is prefix of any other codeword of the code. A one-to-one code is a code that assigns to each source letter a different codeword (notice that in our definition of code, any code is a one-to-one code).

A labeled (binary) tree is a (binary) tree in which each edge is labeled with 0 or with 1 and the two edges from a node to its two children have different labels. A node, except the root, in a labeled tree represents the codeword given by the sequence of labels in the path from the root to that node. Observe that since for our purposes the only important thing about a codeword is its length, we can get rid of the labeling by considering a standard labeling that assigns label 0 to the edge going from a node to its left child, and label 1 to the edge going from a node to its right child. Given a tree, a subset of its nodes not including the root, represents the code consisting of the codewords represented by the nodes of the subset.

It is easy to see that a prefix code can be represented by the set of leaves of a tree, whereas a one-to-one code can be represented by any subset, not including the root, of the nodes of a tree (we are not considering the trivial case of trees consisting of only one node). We can use a binary search tree to define codes. As an example, let $n = 3$ and consider the probability distribution $D = (.35, .20, .10; .10, .05, .05, .15)$. The optimal binary search tree T for D is depicted in fig. 1.

We have that $l(q_1) = 1, l(q_2) = 2, l(q_3) = 3$ and $l(p_0) = 1, l(p_1) = 3, l(p_2) = 3, l(p_3) = 2$. The tree T with the set of its leaves define a prefix code consisting of $n + 1 = 4$ codewords whose lengths are $l(p_0), l(p_1), l(p_2), l(p_3)$ (see fig. 2). The same tree with the set of its internal nodes but the root define a one-to-one code of $n - 1 = 2$ codewords whose lengths are $l(q_2) - 1, l(q_3) - 1$ (see fig. 3). Let T' be the tree consisting of a root with only one child on which is rooted the tree T. The tree T' with the set of its internal nodes but the root define a one-to-one code of $n = 3$ codewords whose lengths are $l(q_1), l(q_2), l(q_3)$ (see fig. 4). The same tree with the set of all its nodes but the root define a one-to-one code of $2n + 1 = 7$ codewords whose lengths are $l(q_1), l(q_2), l(q_3), l(p_0) + 1, l(p_1) + 1, l(p_2) + 1, l(p_3) + 1$ (see fig. 5).

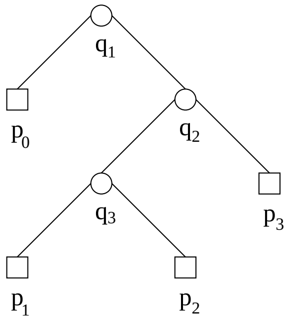
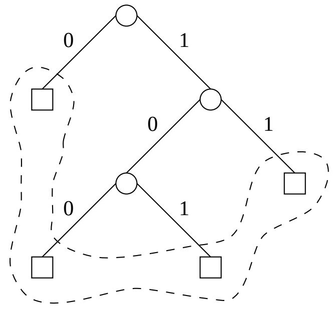
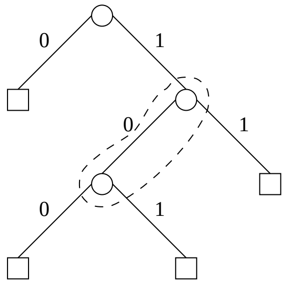


fig. 1



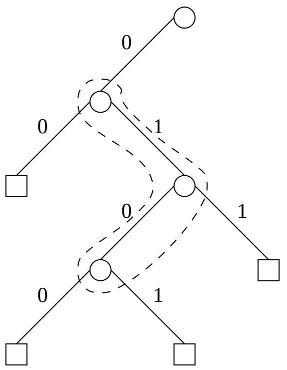
the code is {0,100,101,11}

fig. 2



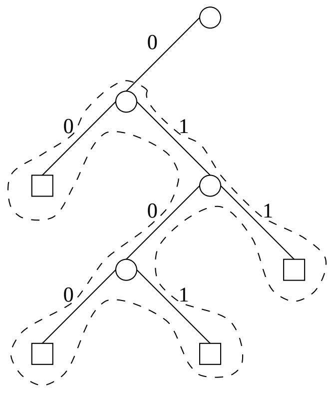
the code is {1,10}

fig. 3



the code is {0,01,010}

fig. 4



the code is {0,00,01,010,0100,0101,011}

fig. 5

Now we recall some lower bounds on the average codeword length of prefix and one-to-one codes.
It is a well-known fact, proved by Shannon, that the average codeword length of a prefix code for a source S must be greater than the entropy of the source S.
SHANNON'S THEOREM. Let L be the average codeword length of a prefix code for a source S

whose entropy is H_S, then

$$L \geq H_S.$$

Let $L_{1:1}$ be the average codeword length of a one-to-one code for a source S of m letters whose entropy is H_S. The following bound is due to Rissanen [7]

$$L_{1:1} \geq H_S - \log\log m. \tag{3}$$

We will use also the following bound due to Leung-Van-Cheong and Cover [6],

$$L_{1:1} \geq H_S - 2\log(H_S + 2). \tag{4}$$

In Section 4 we utilize bounds better than (3) and (4) to improve our results. However, for the sake of simplicity in deriving the bounds we utilize (3) and (4).

Finally, we recall the following results.
KRAFT'S EQUALITY. In any binary search tree we have that $\sum_{k=0}^{n} 2^{-l(p_k)} = 1$.
For the internal nodes of a binary search tree a result which corresponds to the Kraft's equality is the following [1], [2],

$$\sum_{k=1}^{n} 2^{-l(q_k)} \leq \frac{1}{2}\log(n+1). \tag{5}$$

In Section 4 we improve on (5) and using the better bound on the Kraft sum generalized to the internal nodes of a tree, we improve the lower bound on C_{opt} obtained in Section 3 by using the weaker (5).

3 The lower bounds

In this section we derive the bounds. We start with a bound whose proof does not involve the relation between trees and codes. However all the other bounds involve this relation.

Theorem 1 *The cost of any binary search tree satisfies*

$$C \geq H - 1 - Q(\log\log(n+1) - 1). \tag{6}$$

Proof. Recalling the definition of the entropy H and the cost C we have that

$$\begin{aligned} H - C - (\log\log(n+1) - 1)\sum_{k=1}^{n} q_k &= \sum_{k=0}^{n} p_k \log\frac{2^{-l(p_k)}}{p_k} + \sum_{k=1}^{n} q_k \log\frac{2\cdot 2^{-l(q_k)}}{q_k\log(n+1)} \\ &= E[\log(\Lambda)] \end{aligned}$$

where Λ is the random variable which assumes value $2^{-l(p_k)}/p_k$ with probability p_k, for $k = 0, 1, \ldots, n$, and value $2\cdot 2^{-l(q_k)}/q_k\log(n+1)$ with probability q_k, for $k = 1, \ldots, n$. The expected value of Λ is

$$E[\Lambda] = \sum_{k=0}^{n} 2^{-l(p_k)} + \frac{2}{\log(n+1)}\sum_{k=1}^{n} 2^{-l(q_k)}.$$

Using Jensen's inequality $E[\log(\Lambda)] \leq \log(E[\Lambda])$, Kraft's equality and (5), we have that

$$H - C - (\log\log(n+1) - 1)Q \leq \log(E[\Lambda]) \leq 1.$$

■

Bound (6) is better than (2). Indeed, the difference between the former and the latter is $\log e - 1 > 0$. Analogously to bound (2), bound (6) improves on (1) for large values of the entropy (i.e., $H > 2.710 + 2.710Q \log\log(n+1) - 2.710Q$).

Now, following a different reasoning we get new bounds. The technique exploits a binary search tree to define prefix and one-to-one codes. Hence we can use lower bounds on the cost of prefix and one-to-one codes to get lower bounds on the cost of binary search trees.

Theorem 2 *The cost of any binary search tree satisfies*

$$C \geq H - 1 + Q - 2\log(H+2). \tag{7}$$

Proof. Consider the source $\mathcal{S}$ consisting of a set of $2n+1$ letters and the probability distribution D. Let T_{opt} be an optimal binary search tree for D and let C_{opt} be its cost. Construct the following tree T. The tree T consists of a root with only one child on which is rooted T_{opt}. The tree T, with the set of all its nodes but the root, define a one-to-one code for $\mathcal{S}$, whose codewords have lengths $l(q_1), ..., l(q_n), l(p_0)+1, ..., l(p_n)+1$. The average codeword length of such a code is

$$L_{\mathcal{S}} = C_{opt} + 1 - Q.$$

From (4) we have that

$$L_{\mathcal{S}} \geq H_{\mathcal{S}} - 2\log(H_{\mathcal{S}} + 2),$$

and since $H_{\mathcal{S}} = H$ we get the theorem. ∎

The following corollary is immediate.

Corollary 1 *The cost of any binary search tree satisfies*

$$C \geq H - 1 - 2\log(H+2). \tag{8}$$

Bound (8) is better than bound (1) for $H \geq \chi$, where $\chi \simeq 29.714$ is the unique zero of the equation $x - 1 - 2\log(x+2) - x/\log 3 = 0$.
In the following we will denote the binary entropy $H(x, 1-x)$ by $\mathcal{H}(x)$.

Theorem 3 *The cost of any binary search tree satisfies*

$$C_{opt} \geq H - \mathcal{H}(Q) - 2Q\log(H - \mathcal{H}(Q) + 2Q) + 2Q\log Q. \tag{9}$$

Proof. First consider the source $\mathcal{P}$ consisting of a set of $n+1$ letters and probability distribution $(p_0/P, p_1/P, ..., p_n/P)$. Let T_{opt} be an optimal binary search tree for D and let C_{opt} be its cost. This tree with the set of its leaves define a prefix code for $\mathcal{P}$, whose codewords have lengths $l(p_0), l(p_1), ..., l(p_n)$. The average codeword length of such a code is

$$L_{\mathcal{P}} = \frac{1}{P}\sum_{i=0}^{n} p_i l(p_i)$$

whereas the entropy of $\mathcal{P}$ is

$$H_{\mathcal{P}} = \sum_{i=0}^{n} \frac{p_i}{P}\log\frac{P}{p_i} = \log P + \frac{1}{P}\sum_{i=0}^{n} p_i \log\frac{1}{p_i}.$$

The average codeword length $L_{\mathcal{P}}$ satisfies Shannon's theorem, $L_{\mathcal{P}} \geq H_{\mathcal{P}}$, hence

$$\sum_{i=0}^{n} p_i(l(p_i) + \log p_i) \geq P \log P. \tag{10}$$

Now consider the source $\mathcal{Q}$ consisting of a set of n letters and probability distribution $(q_1/Q, \ldots\ldots, q_n/Q)$. Construct the following tree T. The tree T consists of a root with only one child on which is rooted T_{opt}. The tree T, with the set of its internal nodes but the root, define a one-to-one code for $\mathcal{S}$, whose codewords have lengths $l(q_1), \ldots.., l(q_n)$. The average codeword length of such a code is

$$L_{\mathcal{Q}} = \frac{1}{Q} \sum_{i=1}^{n} q_i l(q_i)$$

whereas the entropy of $\mathcal{Q}$ is

$$H_{\mathcal{Q}} = \log Q + \frac{1}{Q} \sum_{i=1}^{n} q_i \log \frac{1}{q_i}.$$

From (4) it follows that $L_{\mathcal{Q}} \geq H_{\mathcal{Q}} - 2\log(H_{\mathcal{Q}} + 2)$, whence

$$\sum_{i=1}^{n} q_i(l(q_i) + \log q_i) \geq Q \log Q - 2Q \log(H_{\mathcal{Q}} + 2).$$

Using above inequality and (10) we have that

$$\begin{aligned} C_{opt} - H &= \sum_{i=0}^{n} p_i(l(p_i) + \log p_i) + \sum_{i=1}^{n} q_i(l(q_i) + \log q_i) \\ &\geq P \log P + Q \log Q - 2Q \log(H_{\mathcal{Q}} + 2) \\ &= -\mathcal{H}(Q) - 2Q \log(H_{\mathcal{Q}} + 2). \end{aligned}$$

Since $H = \mathcal{H}(Q) + QH_{\mathcal{Q}} + PH_{\mathcal{P}} \geq \mathcal{H}(Q) + QH_{\mathcal{Q}}$, we have that $H_{\mathcal{Q}} \leq \frac{H - \mathcal{H}(Q)}{Q}$. Hence the theorem. ∎

When $Q < 1$, bound (9) is clearly better than bound (7) for large values of H. For $Q = 1$ they are equal. Notice that for $Q = 0$ bound (9) is equal to the bound given by Shannon's theorem, as one has to expect, since for $Q = 0$ the cost of a binary search tree is the expected codeword length of a prefix code.

In exploiting the relation between the cost of a binary search tree and the average codeword length of a one-to-one code we used bound (4), which is expressed in terms of the entropy of the source. We can also use bound (3), which is expressed in terms of the entropy of the source and the number of symbols of the source, obtaining bounds that involves H, n and Q.

Theorem 4 *The cost of any binary search tree satisfies*

$$C \geq H - \mathcal{H}(Q) - Q \log \log(n - 1). \tag{11}$$

Proof. Let T_{opt} be an optimal binary search tree for D and let C_{opt} be its cost. Let q_k be the label assigned to the root of T_{opt}. Consider the source $\bar{\mathcal{Q}}$ consisting of a set of $n-1$ letters and probability distribution $(q_1/(Q-q_k), \ldots., q_{k-1}/(Q-q_k), q_{k+1}/(Q-q_k), \ldots., q_n/(Q-q_k))$. The tree T_{opt} with the set of its internal nodes not including the root define a one-to-one code for $\bar{\mathcal{Q}}$. The codeword lengths of such a code are $l(q_1)-1, \ldots., l(q_{k-1})-1, l(q_{k+1})-1, \ldots., l(q_n)-1$. The average codeword length is

$$L_{\bar{\mathcal{Q}}} = \frac{1}{Q-q_k}\sum_{i\neq k} q_i[l(q_i)-1] = \frac{1}{Q-q_k}\sum_{i=1}^{n} q_i l(q_i) - \frac{q_k \cdot l(q_k)}{Q-q_k} - 1.$$

Since $l(q_k)=1$ we get

$$L_{\bar{\mathcal{Q}}} = \frac{1}{Q-q_k}\sum_{i=1}^{n} q_i l(q_i) - \frac{Q}{Q-q_k}$$

The entropy of $\bar{\mathcal{Q}}$ is

$$\begin{aligned} H_{\bar{\mathcal{Q}}} &= \sum_{i\neq k}\frac{q_i}{Q-q_k}\log\frac{Q-q_k}{q_i} \\ &= \frac{1}{Q-q_k}\sum_{i\neq k} q_i\log(Q-q_k) + \frac{1}{Q-q_k}\sum_{i\neq k} q_i\log\frac{1}{q_i} \\ &= \log(Q-q_k) + \frac{1}{Q-q_k}\sum_{i=1}^{n} q_i\log\frac{1}{q_i} - \frac{q_k}{Q-q_k}\log\frac{1}{q_k}. \end{aligned}$$

From (3), the average codeword length $L_{\mathcal{Q}}$ satisfies

$$L_{\bar{\mathcal{Q}}} \geq H_{\bar{\mathcal{Q}}} - \log\log(n-1).$$

By plugging in the above inequality the expressions for $L_{\bar{\mathcal{Q}}}$ and $H_{\bar{\mathcal{Q}}}$ we get

$$\frac{1}{Q-q_k}\sum_{i=1}^{n} q_i l(q_i) - \frac{Q}{Q-q_k} \geq \log(Q-q_k) + \frac{1}{Q-q_k}\sum_{i=1}^{n} q_i\log\frac{1}{q_i} - \frac{q_k}{Q-q_k}\log\frac{1}{q_k} - \log\log(n-1).$$

Rearranging terms we obtain

$$\begin{aligned} \sum_{i=1}^{n} q_i(l(q_i)+\log q_i) &\geq Q + (Q-q_k)\log(Q-q_k) + q_k\log q_k - (Q-q_k)\log\log(n-1) \\ &\geq Q + (Q-q_k)\log(Q-q_k) + q_k\log q_k - Q\log\log(n-1). \end{aligned}$$

It is easy to see that the function $f(x) = (Q-x)\log(Q-x) + x\log x$ is a convex $\cup$ function of x which assumes its minimum at $x = Q/2$. Hence we get

$$\sum_{i=1}^{n} q_i(l(q_i)+\log q_i) \geq Q\log Q - Q\log\log(n-1). \tag{12}$$

Exploiting (12) and (10) we have that

$$\begin{aligned} C_{opt} - H &= \sum_{i=0}^{n} p_i l(p_i) + \sum_{i=1}^{n} q_i l(q_i) + \sum_{i=0}^{n} p_i \log p_i + \sum_{i=1}^{n} q_i \log q_i \\ &= \sum_{i=0}^{n} p_i (l(p_i) + \log p_i) + \sum_{i=1}^{n} q_i (l(q_i) + \log q_i) \\ &\geq P \log P + Q \log Q - Q \log\log(n-1) \end{aligned}$$

Hence the theorem. ■

The difference between (11) and (6) is $1 - \mathcal{H}(Q) - Q + Q[\log\log(n+1) - \log\log(n-1)]$ which is greater than $\Delta(Q) = 1 - \mathcal{H}(Q) - Q$. When $\Delta(Q)$ is positive bound (11) is better than bound (6). The function $\Delta(x) = 1 - \mathcal{H}(x) - x$ is a convex $\cup$ function of x, $0 \leq x \leq 1$, and satisfies $f(0) = 1$, $f(1/2) = -1/2$ and $f(1) = 0$. Hence let δ, $\delta \simeq 0.227$, be the unique zero of the equation $f(x) = 0$, $0 < x < 1$. We have that for $Q \leq \delta$, bound (11) is better than bound (6). For large value of n, (6) is better than (11) for $Q > \delta$. Finally, observe that when $Q \to 0$ bound (11) approaches the limit given by Shannon's theorem, as one has to expect.

Theorem 5 *The cost of any binary search tree satisfies*

$$C \geq H + Q - \mathcal{H}\left(\frac{1}{2 + \log n}\right) - \left(\frac{1 + \log n}{2 + \log n}\right) \log\log(2n). \tag{13}$$

Proof. Let T_{opt} be an optimal binary search tree for D and let C_{opt} be its cost. Let q_k be the label assigned to the root of T_{opt}. Consider the source $\bar{\mathcal{S}}$ consisting of a set of $2n$ letters and probability distribution $(q_1/(1-q_k), ..., q_{k-1}/(1-q_k), q_{k+1}/(1-q_k), ..., q_n/(1-q_k); p_0/(1-q_k), ..., p_n/(1-q_k))$. The tree T_{opt} with the set of all its nodes but the root, define a one-to-one code for $\bar{\mathcal{S}}$, whose codewords have lengths $l(q_1) - 1, ..., l(q_{k-1}), l(q_{k+1}), ..., l(q_n) - 1, l(p_0), ..., l(p_n)$. The average codeword length of such a code is

$$L_{\bar{\mathcal{S}}} = \sum_{i \neq k} \frac{q_i}{1 - q_k}(l(q_i) - 1) + \sum_{i} \frac{p_i}{1 - q_k} l(p_i) = \frac{C_{opt} - Q}{1 - q_k}$$

whereas, since $H = \mathcal{H}(q_k) + (1 - q_k) H_{\bar{\mathcal{S}}}$, the entropy $H_{\bar{\mathcal{S}}}$ of the source $\bar{\mathcal{S}}$ is

$$H_{\bar{\mathcal{S}}} = \frac{H - \mathcal{H}(q_k)}{1 - q_k}.$$

From (3), the average codeword length $L_{\bar{\mathcal{S}}}$ satisfies $L_{\bar{\mathcal{S}}} \geq H_{\bar{\mathcal{S}}} - \log\log(2n)$, that is

$$C_{opt} - Q \geq H - \mathcal{H}(q_k) - (1 - q_k) \log\log(2n).$$

It is easy to see that the function $f(x) = \mathcal{H}(x) + (1 - x) \log\log(2n)$ is a convex $\cap$ function of x and assumes its maximum at $x = \frac{1}{2 + \log n}$. The theorem follows. ■

The bound obtained in the previous theorem improves on (6) for large values of Q. A simple comparison of the two bounds shows that (13) is better than (6) for $Q \geq \phi(n)$, where

$$\phi(n) = \frac{\mathcal{H}\left(\frac{1}{2 + \log n}\right) + \frac{1 + \log n}{2 + \log n} \log\log(2n) - 1}{\log\log(n+1)}.$$

It is easy to see that for large values of n we have that $\phi(n) < 1$. A simple but tedious study of ϕ shows that $\phi(n) < 1$ also for small values of n.

4 Further improvements

In this section we provide improvements of the bounds presented in Section 3.
We can further improve on bound (6) by using a bound on

$$\sum_{k=1}^{n} 2^{-l(q_k)} \tag{14}$$

better than the one provided (5). First observe that above sum reaches the maximum value when all $l(q_k)$ are equal either to $\lfloor \log(n+1) \rfloor$ or to $\lceil \log(n+1) \rceil$. Indeed suppose that there is an internal node at level k which has a child that is a leaf and that there is an internal node at level $j > k+1$ whose children are leaves. The contribution due to the internal node at level j in (14) is 2^{-j}. We can move the subtree rooted at the internal node at level j, rooting it at the external node at level $k+1$, so that the contribution 2^{-j} in (14) becomes $2^{-(k+1)}$, which is greater than 2^{-j}.
Observe that each level which contains only internal nodes gives a contribution in (14) equal to $1/2$, and that there are exactly $\lfloor \log(n+1) \rfloor$ such levels. Each internal node in the unique level with external and internal nodes, gives a contribution in (14) of $2^{-\lceil \log(n+1) \rceil}$. It is easy to see that there are exactly $n+1-2^{\lfloor \log(n+1) \rfloor}$ such internal nodes.
Hence the maximum value of (14) is equal to

$$\frac{1}{2}\lfloor \log(n+1) \rfloor + 2^{-\lceil \log(n+1) \rceil}(n+1-2^{\lfloor \log(n+1) \rfloor}).$$

Therefore the following theorem holds.

Theorem 6 *In any binary search tree the levels of the internal nodes satisfy*

$$\sum_{k=1}^{n} 2^{-l(q_k)} \leq \frac{\lfloor \log(n+1) \rfloor}{2} + \frac{n+1}{2^{\lceil \log(n+1) \rceil}} - \frac{1}{2}$$

and equality holds when the leaves are placed on two consecutive levels.

Above bound is better than (5). In fact it is $< \frac{1}{2}\log(n+1)$ if $n+1$ is not a power of 2, and it is equal to $\frac{1}{2}\log(n+1)$ if $n+1$ is a power of 2. Utilizing this bound in Theorem 1, we can improve on (6).

We can also get an improvement of bounds (11) and (13). In fact, in deriving these bounds we utilized (3). We can use the following bound, due to Rissanen [7], that gives a sharper bound on the average codeword length $L_{1:1}$ of a one-to-one code for a source S of m letters whose entropy is H_S

$$L_{1:1} \geq H_S - \log \alpha(m)$$

where $\alpha(m) = k(m) - 1 + r(m)2^{-k(m)}$ and $k(m)$ is the maximum integer such that $r(m) = m - 2^{k(m)} + 2$ is positive. Moreover $\alpha(m) < \log m$.

We can also improve on bounds (7) and (9) by using a bound better than (4). Actually, Verriest [10] proved that the average codeword length $L_{1:1}$ of a one-to-one code for a source S whose entropy is H_S, is greater than or equal to the value L_{min} given by the equation

$$H_S = L_{min}\left(1 + \mathcal{H}\left(\frac{1}{L_{min}}\right)\right)$$

and this limitation is the best possible on $L_{1:1}$ in terms of H_S only. By using the bound $L_{1:1} \geq L_{min}$ we get a bound better than (4), and thus we can improve on (7) and (9).

However all the improvements remarked in this section are not very significant and the expressions of the stronger bounds are quite complicate. Hence, for sake of simplicity, we presented in Section 3 the slightly weaker bounds.

References

[1] R. Ahlswede and I. Wegener, *Search problems*, (J.Wiley & Sons, 1987).

[2] M. Aigner, *Combinatorial search*, (J.Wiley & Sons, 1988).

[3] P.J. Bayer, Improved bounds on the cost of optimal and balanced binary search trees, M.Sc. Thesis, Mass.Inst.of Tech., Cambridge, MA, 1975.

[4] R. De Prisco and A. De Santis, On binary search trees, *Information Processing Letters*, **45**, (Apr 1993), pp. 249–253.

[5] E.N. Gilbert and E.F. Moore, Variable-length binary encodings, *Bell System Tech J.* **38** (1959), pp. 933—968.

[6] S.K. Leung-Yan-Cheong and T.M. Cover, Some equivalences between Shannon entropy and Kolmogorv complexity, *IEEE Trans. Inf. Theory*, **24** (May 1978), pp. 331–338.

[7] J. Rissanen, Tight lower bounds for optimum code length, *IEEE Trans. Inf. Theory*, **18** (Mar 1982), pp. 348–349.

[8] K. Mehlhorn, Nearly optimal binary search trees, *Acta Informatica* **5**, (1975), pp. 287–295.

[9] K. Mehlhorn, A best possible bound for the weighted path length of binary search trees, *SIAM J.Comput.* **2**, (1977), pp. 235–239.

[10] E.I. Verriest, An achievable bound for optimal noiseless coding of a random variable, *IEEE Trans. Inf. Theory*, **32** (Jul 1986), pp. 592–594.