Correspondence

The Estimated Cost of a Search Tree on Binary Words

Alexey Fedotov and Boris Ryabko

***Abstract*—The problem of constructing a binary search tree for a set of binary words has wide applications in computer science, biology, mineralogy, etc. Shannon considered a similar statement in his optimal coding theorem. It is NP-complete to construct a tree of minimum cost [4]; therefore, the problem arises of finding simple algorithms for constructing nearly optimal trees. We show in this correspondence that there is a simple algorithm for constructing search trees sufficiently close to the optimal tree on average. By means of this algorithm we prove that for the optimal tree the average number of bits to be checked is near to its natural lower bound, i.e., the binary logarithm of the number of given words: their difference is less than 1.04.**

***Index Terms*—Binary tree, search tree, Shannon theorem.**

I. Introduction

The algorithm of search in a given set of binary words can be represented as a binary tree. The leaves of this tree correspond to the given words, and the internal nodes of the tree correspond to the indexes of positions to be checked. To identify a word we have to move along the edges of the tree from the root to one of the leaves. The direction of movement in nodes is selected depending on the corresponding bit in the given word.

Fig. 1 demonstrates two search trees in the set of words $\{000, 010, 110, 111\}$. We will explain how we could perform a search using the upper tree. Initially, the first position of the examined word is checked. If it is equal to 0, then we go to the left branch of the tree and check the second position. Otherwise, we check the third position of a word. Thus, two checks are always sufficient to determine a given word.

The binary search trees are commonly used in computer science, information theory, biology, mineralogy, etc. Therefore, the problem of constructing such trees has attracted the attention of many authors [1]–[3]. Naturally, there is a problem of obtaining a search tree for a given set of binary words of the same length. The constructed tree is assessed by the cost of search, which is defined as the average number of bits required for identifying a word. It is natural to look for a tree with the minimum average number of bits to be checked, the so-called optimal tree. This problem is close to the problem of constructing the code with the minimum average codeword length. The latter is well known in information theory. For example, the average number of bits to be checked for identifying a word in the upper tree in Fig. 1 is equal to 2 and in the lower tree it is equal to 2.25. It is well known in information theory that the cost of each tree cannot be less than $\log_2 m$, where m is the number of given words. Thus, the upper tree is optimal.

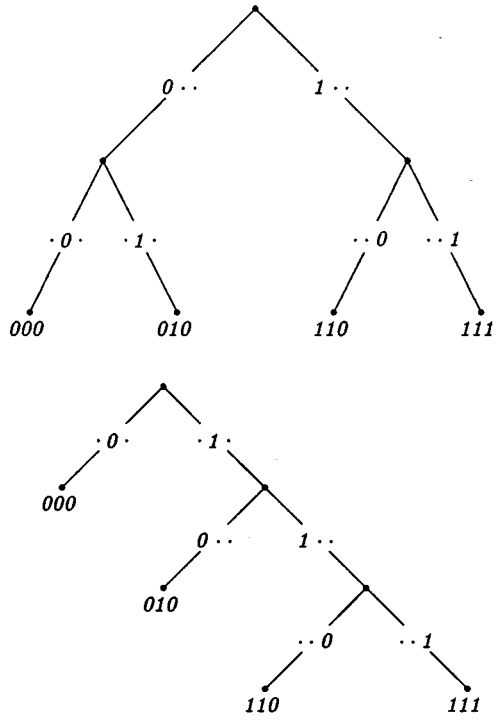

Fig. 1. Example of search trees with different costs.

The problem of finding an optimal search tree for a given set of words is known to be NP-hard [4]. Informally, it means that this problem is hard to compute when the quantity of words is large enough. Therefore, the problem of discovering a fast algorithm that constructs nearly optimal search trees emerges.

We show in this correspondence that even a quite simple randomized algorithm which constructs a search tree, gives as a rule a nearly optimal result. More precisely, on average (on the whole set of initial data), the cost of search using a tree constructed by the above algorithm does not exceed $\log_2 m + 1.04$. Hence, the cost of search by means of an optimal tree does not exceed this bound either. Thus, the Shannon theorem which estimates the cost of coding is fulfilled on average when a code tree is constructed from a random set of words. The constant 1.04 is an additional cost (instead of 1 in the Shannon estimate). This good upper bound is obtained from the observation that a typical check of a bit in a random set of words divides this set into two parts of nearly equal size. Hence the number of checks is near $\log_2 m$.

Using the main result, we find the estimate of a cost of a search tree which holds on almost every set of initial data.

II. The Main Result

Assume a set of m binary words of length n $(m \geq 0, n \geq 1)$ is given.

Manuscript received November 9, 1999; revised June 28, 2000. This work was supported by the RFBR under Grant 98-01-00772. The material in this correspondence was presented in part at the IEEE International Symposium on Information Theory, Sorrento, Italy, June 25–30, 2000.

A. Fedotov is with the Department of Informational Technologies, Institute of Computational Technologies, Novosibirsk-90, Russia (e-mail: lesha@adm.ict.nsc.ru).

B. Ryabko is with the Department of Applied Mathematics and Cybernetics, Siberian State University of Telecommunications and Information Sciences, Novosibirsk-102, Russia (e-mail: ryabko@neic.nsk.su).

Communicated by M. Weinberger, Associate Editor for Source Coding.

Publisher Item Identifier S 0018-9448(01)00375-3.

Let $C(L) = \frac{1}{m}\sum_{i=1}^{m} L_i$, where L_i is the number of bits required for identification of the ith word. We call $C(L)$ the cost of a search tree L, i.e., the average amount of bits required for identifying a word by the tree L.

We denote by $\mathcal{S}_{n,m}$ the set of initial data, or, in other words, the collection of all the sets of m binary words of length n. It is clear that the requirement can be met only when $n \geq \log_2 m$.

We assume now that an algorithm F builds a tree $F(S)$ from a set $S \in \mathcal{S}_{n,m}$. We will further consider randomized algorithms, and so it will be convenient to denote the expectation of the cost of the tree $C(F(S))$ (with respect to the measure induced by the algorithm) by the average cost $\bar{C}(F(S))$ of algorithm F on the input S. Let us define now the average cost $t_{n,m}(F)$ of the algorithm F as the arithmetical mean of average costs of the algorithm F calculated on all inputs from the collection $\mathcal{S}_{n,m}$

$$t_{n,m}(F) = \frac{1}{\operatorname{Card}\mathcal{S}_{n,m}} \sum_{S\in\mathcal{S}_{n,m}} \bar{C}(F(S)),$$

where $\operatorname{Card}\mathcal{S}_{n,m}$ means the cardinality of the set $\mathcal{S}_{n,m}$.

We now consider the (possibly) simplest randomized algorithm constructing a search tree, which will be denoted by R. It can be described as follows.

Description of the Algorithm R: This algorithm makes a binary search tree from an arbitrary set of m binary words of length n. If the given set contains only one word then the algorithm returns the simplest tree consisting of one leaf and stops.

Otherwise, the randomly chosen position is put into correspondence with the root of the tree. This check divides the entire set of words into two parts. For each of these parts the search tree is constructed by the same method.

The main result of this correspondence is the following theorem.

Theorem 1: For the average cost of the algorithm R the following inequality holds:

$$t_{n,m}(R) \leq \log_2 m + \frac{29}{28} - \frac{\log_2(2m)}{m}. \tag{1}$$

From this result the following corollary is immediate.

Corollary 1: Let F_{opt} be the algorithm building an optimal tree for each data set. Then

$$t_{n,m}(F_{\text{opt}}) \leq \log_2 m + \frac{29}{28} - \frac{\log_2(2m)}{m}.$$

The following corollary gives an estimate for the cost of the search tree constructed by R for almost all data sets. Moreover, we obtain the same estimate for the cost of the optimal search tree.

Corollary 2: Assume that all sets S from the collection of initial data $\mathcal{S}_{n,m}$ $(m \geq 2, n \geq \log_2 m)$ have the same probability. Then, for every $\epsilon > 0$ the following inequality holds:

$$P\{S{:}\bar{C}(R(S)) < (1+\epsilon)\log_2 m\} \geq 1 - \frac{29/28}{\epsilon \log_2 m}.$$

Proof: This corollary is an obvious consequence of the Markov inequality [3]. □

III. Proofs

We start by establishing the recurrence that holds for the average cost $t_{n,m}(R)$ of the algorithm R. Then, we find a number of exact values of $t_{n,m}(\cdot)$ that are further employed. In what follows we show two technical lemmas and use them to prove Theorem 1. At the end we present a bulky proof of the first technical lemma.

We first obtain the recurrence for $t_{n,m}(R)$.

Lemma 1: For the average cost of the algorithm R we have

$$t_{n,m}(R) = 1 + \sum_{k=0}^{m} c_{n,m,k}\left(\frac{k}{m}t_{n-1,k}(R) + \frac{m-k}{m}t_{n-1,m-k}(R)\right) \tag{2}$$

where $c_{n,m,k}$ is the probability that exactly k words from the given m words begin with 1. This probability is given in (3) at the bottom of this page. (Here and throughout we assume that $t_{n,0} \equiv 0$.)

Proof: We begin with the proof of (3). Denote by $\mathcal{S}^k_{n,m}$ the subcollection of $\mathcal{S}_{n,m}$ which consists of the sets containing exactly k words beginning with 1. By definition

$$c_{n,m,k} = \operatorname{Card}\mathcal{S}^k_{n,m} \Big/ \operatorname{Card}\mathcal{S}_{n,m}.$$

Obviously, $\operatorname{Card}\mathcal{S}_{n,m} = \binom{2^n}{m}$. Also, if $m - 2^{n-1} \leq k \leq 2^{n-1}$, then the set $\mathcal{S}^k_{n,m}$ contains $\binom{2^{n-1}}{k}\binom{2^{n-1}}{m-k}$ elements, or otherwise this set is empty. Thereby we obtain (3).

Consider the algorithm R applied to the input S. We suppose that this set contains k words with 1 and $m-k$ words with 0 in the first chosen position. In this situation, the cost of the algorithm is equal to

$$1 + \frac{k}{m}\bar{C}(R(T_1S)) + \frac{m-k}{m}\bar{C}(R(T_0S))$$

where T_0S and T_1S are subsets of the set S consisting of words of S with 0 and 1 in the selected position, respectively. Moreover, the selected bit is excluded from these words. By averaging this equality over $\mathcal{S}_{n,m}$ we obtain the following:

$$t_{n,m}(R) = 1 + \sum_{k=0}^{m} \frac{\operatorname{Card}\mathcal{S}^k_{n,m}}{\operatorname{Card}\mathcal{S}_{n,m}}\left(\frac{k}{m}\sum_{S\in\mathcal{S}^k_{n,m}} \frac{\bar{C}(R(T_1S))}{\operatorname{Card}\mathcal{S}^k_{n,m}} + \frac{m-k}{m}\sum_{S\in\mathcal{S}^{m-k}_{n,m}} \frac{\bar{C}(R(T_1S))}{\operatorname{Card}\mathcal{S}^{m-k}_{n,m}}\right).$$

It remains to note that

$$\sum_{S\in\mathcal{S}^k_{n,m}} \frac{\bar{C}(R(T_1S))}{\operatorname{Card}\mathcal{S}^k_{n,m}} = t_{n-1,k}(R).$$

The proof is complete. □

In the following proofs, we use a number of exact values of the average cost $t_{n,m}(\cdot)$, which are shown in Table I. Here, we explain the main idea of the calculation. For pairs n, m in Table I the situation is considerably simpler than in Fig. 1.

For these cases all search trees are isomorphic. Hence their average cost is equal and independent of the constructing algorithm.

Now we estimate $t_{n,4}(R)$ using Lemma 1.

$$c_{n,m,k} = \begin{cases} \binom{2^{n-1}}{k}\binom{2^{n-1}}{m-k} \Big/ \binom{2^n}{m}, & \text{when } m - 2^{n-1} \leq k \leq 2^{n-1} \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Theorem 2: For the average cost of the trees constructed by the algorithm R, for search in the set of four elements $(n \geq 2)$

$$t_{n,4}(R) \leq \frac{16}{7}. \tag{4}$$

Proof: We prove that $t_{n,4} \leq 16/7$ for $n \geq 2$ by induction on n. The base of induction is obvious: $t_{2,4} = 2 < 16/7$. To prove the step we have to show that the inequality (4) remains valid when n increases.

Evidently, (2) can be rewritten as

$$t_{n,m}(R) = 1 + \sum_{k=0}^{m} c_{n,m,k} \frac{2k}{m} t_{n-1,k}(R). \tag{5}$$

Now we can represent $t_{n,4}(R)$ using (5). Replace $t_{n-1,0}(R)$, $t_{n-1,1}(R)$, $t_{n-1,2}(R)$, and $t_{n-1,3}(R)$ with the values taken from Table I and $t_{n-1,4}(R)$ with the upper bound given by the induction hypothesis. Then the induction step can be proved by the following inequality:

$$1 + \frac{2}{2} c_{n,4,2} + \frac{3}{2} c_{n,4,3} \frac{5}{3} + 2 c_{n,4,4} \frac{16}{7} \leq \frac{16}{7}$$

which is a simple consequence of the inequality $-27 \cdot 2^{n-1} + 42 \leq 0$. The latter is valid when $n \geq 3$. The proof is complete. □

Now we give two technical lemmas to be used in the proof of Theorem 1. The first result establishes a polynomial upper bound for some clumsy function.

Lemma 2: Let $m \geq 5$. Consider the function

$$f(k) = \frac{k}{m} \log_2(2k) \left(1 - \frac{1}{k}\right) + \frac{m-k}{m} \log_2(2(m-k)) \left(1 - \frac{1}{m-k}\right)$$

defined on the segment $[1, m-1]$. Then for every k in $[1, m-1]$ the inequality $f(k) \leq A - Bk(m-k)$ holds, where

$$A = \log_2(2m)\left(1 - \frac{1}{m}\right)$$
$$B = \frac{4}{m^3}(m + \log_2 m - 1).$$

The proof is relegated to the end of the correspondence.

Now we prove a combinatorial result that will be used to simplify weighted sums.

Lemma 3: The following identity holds:

$$\sum_{k=0}^{m} c_{n,m,k} k(m-k) = \frac{2^n}{2^n - 1} \frac{m(m-1)}{4} \tag{6}$$

where $c_{n,m,k}$ are defined in (3).

Proof: To prove this statement we equate coefficients of x^{m-2} in the polynomial identity

$$(x-1)^{2^{n-1}-1}(x-1)^{2^{n-1}-1} = (x-1)^{2^n-2}.$$

Equation (6) clearly follows from the above equation. The proof is complete. □

We can now prove Theorem 1.

Proof: We prove this theorem by induction on n. Inequality (1) is readily checked for $m = 1, 2, 3$ using the values of $t_{n,m}(\cdot)$ from Table I. In the same way we check the inequality for $m = 4$, using the upper bound for $t_{n,4}(R)$ given by Theorem 2.

TABLE I
SOME VALUES OF $t_{n,m}(\cdot)$

$t_{n,m}(\cdot)$	n			
m	1	2	...	n
0	0	0	...	0
1	0	0	...	0
2	1	1	...	1
3	—	5/3	...	5/3
4	—	2	...	...
⋮	⋮	⋮	⋱	⋮
$2^n - 2$	0	1	...	$n - 2/m$
$2^n - 1$	0	5/3	...	$n - 1/m$
2^n	1	2	...	n

The preceding reasoning establishes the base of induction for $n = 1, 2$.

We now prove the step. The induction hypothesis implies that for all k satisfying $1 \leq k \leq \log_2(n-1)$ inequality (1) holds for $t_{n-1,k}(R)$. We show that inequality (1) holds for all m satisfying $5 \leq m \leq \log_2 n$.

We first establish the following inequality for all k from 0 to m:

$$\frac{k}{m} t_{n-1,k}(R) + \frac{m-k}{m} t_{n-1,m-k}(R) \leq A_1 - Bk(m-k) \tag{7}$$

where

$$A_1 = \frac{1}{28} + \log_2(2m)\left(1 - \frac{1}{m}\right)$$
$$B = \frac{4}{m^3}(m + \log_2 m - 1).$$

To prove it we note that $t_{n-1,0}(R) = 0$, $t_{n-1,m}(R) \leq A_1$. Therefore, for $k = 0$ and $k = m$, (7) can be easily checked. For k ranging from 1 to $m-1$ (7) can be verified by substituting the upper bound (1) for $t_{n-1,k}(R)$ and applying Lemma 2 to the result.

We conclude that (2) in Lemma 1 gives the following upper bound for $t_{n,m}(R)$:

$$t_{n,m}(R) \leq 1 + \sum_{k=0}^{m} c_{n,m,k}(A_1 - Bk(m-k)).$$

Using Lemma 3, we can reduce the right-hand side of this equation to the following form:

$$t_{n,m}(R) \leq 1 + A_1 - B \frac{2^n}{2^n - 1} \cdot \frac{m(m-1)}{4}.$$

To complete the proof it remains to show that the following inequality holds:

$$A_1 - B \frac{2^n}{2^n - 1} \cdot \frac{m(m-1)}{4} \leq \log_2 m + \frac{1}{28} - \frac{\log_2(2m)}{m}.$$

This is easily derived by multiplying the obvious inequalities $1 < 2^n/(2^n - 1)$ and $m^2 < (m-1)(m - 1 + \log_2 m)$ for $m \geq 5$. The proof is complete. □

To conclude, we prove Lemma 2. The proof is similar to that of the well-known [2] upper bound for the entropy $H(p) \leq 4p(1-p)$.

Proof of Lemma 2: Denote by $g(k)$ the difference $A - Bk(m-k) - f(k)$. To clarify the proof we depict in Fig. 2 the graph of $g(k)$ for $m = 100$. Now we show that when $m \geq 5$ the function $g(k)$ is nonnegative on the segment $[1, m-1]$. We prove the following facts on $g(k)$.

1) The function $g(k)$ is an even function about the midpoint of the segment $[1, m-1]$: i.e. $g(k) = g(m-k)$. The proof is evident.

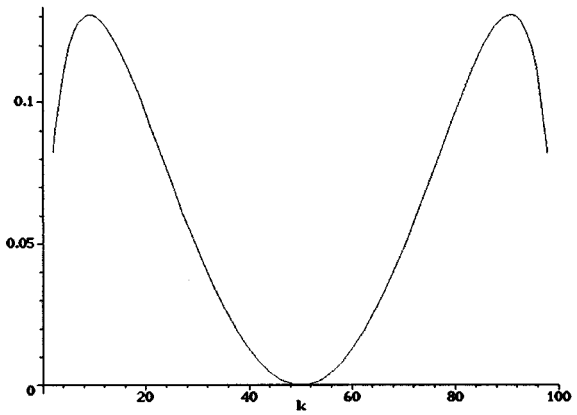

Fig. 2. The graph of $g(k)$ for $m = 100$.

2) At the midpoint of the segment $[1, m-1]$ the function $g(k)$ vanishes. The proof is a direct calculation of $g(m/2)$.

3) At the endpoints of the segment $[1, m-1]$ the function $g(k)$ is positive.
Notice that $\log_2 e > 7/5$. Then the inequality $m(\log_2 e - 1) > 2$ holds for $m \geq 5$. By simple transformations we obtain the following inequalities:

$$-2m + 2 + m(\log_2 e - 1) > -2m + 4,$$
$$2(m-2) > 3m - 2 - m\log_2 e.$$

The inequality $\log_2 m > 2$ implies

$$(m-2)\log_2 m > 3m - 2 - m\log_2 e.$$

The latter is tantamount to

$$m(m-2)\log_2 e + (m-2)^2 \log_2 m + m^2 > 4(m-1)^2$$

Clearly, the inequality $-\log_2(1-1/m) \geq \log_2 e/m$ holds for $m \geq 1$. Therefore, we can convert the previous expression into the following form:

$$-m^2(m-2)\log_2\left(1 - \frac{1}{m}\right) + (m-2)^2\log_2 m + m^2 > 4(m-1)^2$$
$$-f(1) + \frac{m-2}{m}\log_2(2m) + \frac{1}{m}\log_2 m + \frac{1}{m} > (m-1)B.$$

This proves that $g(m-1) = g(1) = A - B(m-1) - f(1) > 0$ for $m \geq 5$. The proof is complete.

4) The second derivative of $g(k)$ decreases strictly to $-\infty$ from the midpoint of the interval $(0, m)$ to its endpoints.
To prove it we calculate $g''(k)$

$$g''(k) = 2B - \frac{\log_2 e}{m}\left(\frac{1}{k} + \frac{1}{k^2} + \frac{1}{m-k} + \frac{1}{(m-k)^2}\right).$$

Denote $(k/m - 1/2)^2$ by d. In this case, the expression in brackets can be transformed to the following form:

$$\frac{1}{m(1/4-d)} + \frac{d + 1/2}{m^2(1/4-d)^2}.$$

This expression increases strictly when d goes from 0 to $1/4$, i.e. k goes from the midpoint of the interval to its endpoints, clearly implying the statement. The proof is complete.

5) The second derivative $g''(m/2)$ is positive at the midpoint of the segment $[1, m-1]$.
We can easily calculate it as

$$g''(m/2) = 2B - \frac{\log_2 e}{m}\left(\frac{4}{m} + \frac{8}{m^2}\right)$$
$$= \frac{4}{m^3}(2(m - 1 + \log_2 m) - \log_2 e(1+m)) > 0$$

since for $m \geq 4$ the following chain of inequalities holds:

$$2(m - 1 + \log_2 m) \geq 2(m - 1 + 2) \geq \log_2 e(1+m).$$

Now we have to show that a minimum of $g(k)$ on the segment $[1, m-1]$ is reached at one of the three points $\{1, m/2, m-1\}$. Since the function $g(k)$ is nonnegative at these points, it is positive over the whole segment $[1, m-1]$, as was to be shown.

The decrease of the second derivative of $g(k)$ from the positive value at the midpoint of the interval $(0, m)$ to $-\infty$ when k approaches its endpoints implies the existence of some δ that satisfies $g''(m/2 \pm \delta) = 0$. Moreover, this derivative must be positive on the open interval $(m/2 - \delta, m/2 + \delta)$ and it must be negative outside the closed interval.

Then we obtain that $g(k)$ is convex upwards on the segment $[m/2 - \delta, m/2 + \delta]$. Since $g(k)$ is symmetric about the midpoint of this segment, $g(k)$ attains a minimum at this midpoint. If $m/2 - \delta \leq 1$, then the previous reasoning completes the proof. Otherwise, we note that function $g(k)$ is convex downwards on the segments $[1, m/2-\delta]$ and $[m/2+\delta, m-1]$. Hence, it reaches a minimum at one of their endpoints. Therefore, a minimum of $g(k)$ on the segment $[1, m-1]$ is reached at one of the following points: $\{1, m/2, m-1\}$. The proof is complete. □

ACKNOWLEDGMENT

The authors wish to thank M. A. Alekseyev for useful discussions and to the anonymous referee for helpful remarks.

REFERENCES

[1] D. Knuth, *The Art of Computer Programming.* Reading, MA: Addison-Wesley, 1973, vol. 3.
[2] R. Ahlswede and I. Wegener, *Suchprobleme*. Stuttgart, Germany: Teubner, 1979.
[3] R. Krichevsky, *Universal Compression and Retrieval.* Norwell, MA: Kluwer, 1994.
[4] M. Garey and D. Johnson, *Computers and Intractability*. CA: Freeman, 1979.