Best Possible Bounds on the Weighted Path Length of Optimum Binary Search Trees

by

Kurt Mehlhorn

Abstract :

We derive upper and lower bounds for the weighted path length P_{opt} of optimum binary search trees. In particular,

$$1/\log 3\ H \leq P_{opt} \leq 2 + H$$

where H is the entropy of the frequency distribution. We also present an approximation algorithm which constructs nearly optimal trees.

I. Introduction

"One of the popular methods for retrieving information by its 'name' is to store the names in a binary tree. We are given n names $B_1, B_2, \dots, B_n$ and 2n+1 frequencies $\beta_1, \dots, \beta_n, \alpha_o, \dots, \alpha_n$ with $\Sigma\beta_i + \Sigma\alpha_j = 1$. Here β_i is the frequency of encountering name B_i, and α_j is the frequency of encountering a name which lies between B_j and B_{j+1}, (a name in the interval (B_j, B_{j+1})) α_o and α_n have obvious interpretations". ([5]).

Fachbereich 1o
Angewandte Mathematik und Informatik
Universität des Saarlandes, D 66 Saarbrücken

A binary search tree T is a tree with n interior nodes (nodes having two sons), which we denote by circles, and n + 1 leaves, which we denote by squares. The interior nodes are labelled by the B_i in increasing order from left to right and the leaves are labelled by the intervals (B_j, B_{j+1}) in increasing order from left to right. Let b_i be the distance of interior node B_i from the root and let a_j be the distance of leaf (B_j, B_{j+1}) from the root. To retrieve a name X, $b_i + 1$ comparisons are needed if $X = B_i$ and a_j comparisons are required if $B_j < X < B_{j+1}$. Therefore we define the weighted path lenght of tree T as :

$$P = \sum_{i=1}^{n} \beta_i \ (b_i+1) + \sum_{j=o}^{n} \alpha_j \ a_j$$

It is equal to the expected number of comparisons needed to retrieve a name.

The following two problems are among the most important in this area ([5]).

a) Prove good lower and upper bounds for the weighted path length of optimum binary search trees, i.e. the trees with minimal weighted path length. Such bounds would provide us with a simple a-priori test for the performance of binary search trees.

b) Design efficient algorithms for constructing optimal (or nearly so) binary search trees.

In this paper, we attempt to solve both problems.

II. Upper Bounds

In this section, we will show that $1 + \sum \alpha_j + H$ -- $H = - \sum \beta_i \log \beta_i - \sum \alpha_j \log \alpha_j$ is the entropy of the frequency distribution -- is an upper bound on the weighted path length P_{opt} of the optimum binary search tree. Furthermore this bound is best possible among the bounds of the form
$c_1 \sum \beta_i + c_2 \sum \alpha_j + c_3 \cdot H$.

We prove the upper bound by describing and analyzing an approximation algorithm. This algorithm constructs binary search trees in a top-down fashion. It uses bisection on the set

$$\{ s_i \; ; \; s_i = \sum_{p=o}^{i-1} (\alpha_p + \beta_p) + \beta_i + \alpha_i/2 \text{ and } o \le i \le n \},$$ i.e.

the root k is determined such that $s_{k-1} \le 1/2$ and $s_k \ge 1/2$. It proceeds then recursively on the subsets $\{ s_i ; i \le k - 1 \}$ and $\{ s_i ; i \ge k \}$.

The main program

<u>begin</u>

let $s_i \leftarrow \sum_{p=o}^{i-1} (\alpha_p + \beta_p) + \beta_i + \alpha_i/2$ for $o \le i \le n$;

construct-tree (0, n, 0, 1)

<u>end</u>

uses the recursive procedure construct-tree.

construct-tree (i, j, cut, ℓ) ;

<u>comment</u> we assume that the actual parameters of any call of construct-tree satisfy the following conditions.

(1) i and j are integers with $o \le i < j \le n$,

(2) ℓ is an integer with $\ell \geq 1$,

(3) $\text{cut} = \sum_{p=1}^{\ell-1} x_p\, 2^{-p}$ with $x_p \in \{0, 1\}$ for all p,

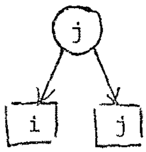
(4) $\text{cut} \leq s_i \leq s_j \leq \text{cut} + 2^{-\ell+1}$.

A call construct-tree (i, j, -, -,) will construct a binary search tree for the nodes i+1 ,..., j and the leaves $\boxed{i}$,..., $\boxed{j}$;

begin

if i + 1 = j (case A)

then return the tree



else comment we determine the root so as to bisect the interval

(cut, cut + $2^{-\ell+1}$)

begin

determine k such that

(5) $i < k \leq j$

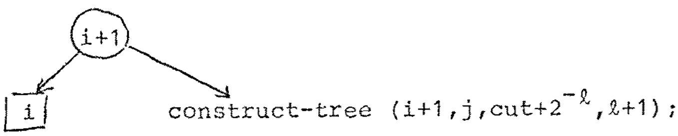
(6) $k = i + 1$ or $s_{k-1} \leq \text{cut} + 2^{-\ell}$

(7) $k = j$ or $s_k \geq \text{cut} + 2^{-\ell}$

comment k exists because the actual parameters are supposed to satisfy condition (4);

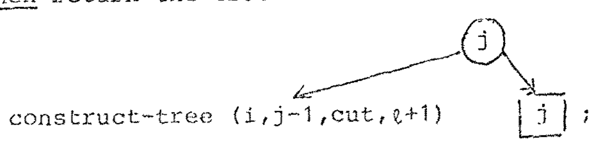
if k = i + 1 (case B)

then return the tree



if k = j (case C)

then return the tree



<u>if</u> $i + 1 < k < j$ (case D)

<u>then</u> return the tree

$$\begin{tikzcd} & k \arrow[dl] \arrow[dr] & \\ \text{construct-tree } (i,k-1,cut,\ell+1) & & \text{construct-tree } (k,j,cut+2^{-\ell},\ell+1) \end{tikzcd}$$

<u>end</u>

<u>end</u>.

<u>Lemma :</u>

The approximation algorithm constructs a binary search tree whose weighted path length P_{approx} is bounded above by

$$1 + \sum \alpha_j + H.$$

The algorithm can be implemented to work in O (n log n) units of time and O (n) units of space.

<u>proof :</u>

We state several simple facts.

<u>Fact 1 :</u>

If the actual parameters of a call construct-tree (i, j, cut, ℓ) satisfy conditions (1) to (4) and $i + 1 \neq j$ then a k satisfying conditions (5) to (7) exists and the actual parameters of the recursive calls of construct-tree iniated by this call again satisfy conditions (1) to (4).

<u>Fact 2 :</u>

The actual parameters of every call of construct-tree satisfy conditions (1) to (4) (if the arguments of the top-level call do).

We say that node h (leaf $\boxed{h}$ respectively) is constructed by call construct-tree (i, j, cut, ℓ) if $h = j$ ($h = i$ or $h = j$) and case A is taken or if $h = i + 1$ ($h = i$) and case B is taken or if $h = j$ ($h = j$) and case C is taken or if $h = k$ and case D is taken. Let b_i be the depth of node i and let a_j be the depth of leaf $\boxed{j}$ in the tree returned by the call construct-tree (o, n, o, 1).

Fact 3 :

If node h (leaf $\boxed{h}$) is constructed by the call construct-tree (i, j, cut, ℓ) then $b_h + 1 = \ell$ ($a_h = \ell$).

Fact 4 :

If node h (leaf $\boxed{h}$) is constructed by the call construct-tree (i, j, cut, ℓ) then $\beta_h \leq 2^{-\ell+1}$ ($\alpha_h \leq 2^{-\ell+2}$).

Fact 5 :

The weighted path length P_{approx} of the tree constructed by the approximation algorithm is bounded above by $1 + \sum\alpha_j + H$.

We sketch now an efficient implementation of our approximation algorithm. The complexity of the algorithm is determined by the complexity of the search for k. If we search for k simultaneously from both ends, i.e. try $k = i + 1$, $k = j$, $k = i + 2$, $k = j - 1$, ... successively, then the complexity of this search is O (min ($k - i$, $j - k + 1$)). Hence we get the following recurrence relation for the complexity of construct-tree (as a function of $j - i$).

$$(*) \quad T(m) \le \begin{cases} 0 & \underline{\text{if}}\ n = 0 \text{ (by definition)} \\ c_1 & \underline{\text{if}}\ n = 1 \\ \max\limits_{0<k<m/2} \left[T(k) + T(m-k+1) + c_2\,(k+1)\right] & \text{otherwise} \end{cases}$$

for some constants c_1, c_2.

<u>Fact 7 :</u>

The recurrence relation ($*$) has a solution $T(n) \le O(n \log n)$.

<u>Fact 8 :</u>

The approximation algorithm can be implemented to work in $O(n \log n)$ units of time and $O(n)$ units of space.

q. e. d.

<u>Theorem 1 :</u>

Let $\alpha_0, \beta_1, \alpha_1, \ldots, \beta_n, \alpha_n$ be any frequency distribution, let P_{opt} be the weighted path length of the optimum binary search tree for this distribution, let P_{approx} be the weighted path length of the tree constructed by the approximation algorithm, and let $H = -\sum \beta_i \log \beta_i - \sum \alpha_j \log \alpha_j$ be the entropy of the frequency distribution. Then

$$P_{opt} \le P_{approx} \le 1 + \sum \alpha_j + H.$$

Furthermore, this upper bounds is best possible in the following sense : If $c_1 \sum \beta_i + c_2 \sum \alpha_j + c_3 \cdot H$ is an upper bound on P_{opt} then $c_1 \ge 1$, $c_2 \ge 2$, and $c_3 \ge 1$.

<u>proof :</u>

The first part of the theorem follows from the preceding lemma. The second part is proven by exhibiting suitable frequency distributions.

$c_1 \geq 1$: Take $n = 1$, $\alpha_o = \alpha_1 = o$ and $\beta_1 = 1$.

$c_2 \geq 2$: Take $n = 2$, $\alpha_o = \alpha_2 = \beta_1 = \beta_2 = o$, $\alpha_1 = 1$.

$c_3 \geq 1$: Take $n = 2^k - 1$, $\beta_i = o$ for all i and $\alpha_j = 2^{-k}$ for all j. It is easy to see that the complete binary tree is the optimal binary search tree for this distribution. Thus

$$H = \log n + 1 = k = \sum_{\text{leaves}} 1/2^k \cdot k = P_{opt}.$$

q. e. d.

E.N. Gilbert and E.F. Moore (1) proved this theorem in the special case that all internal nodes have weight zero (i.e. $\beta_i = o$ for all i). Their proof suggest the approximation algorithm which we presented above. Other " rules of thumb " are discussed in [7, 8] ; we prove in [7] that the strategy " choose the root so as to equalize the total weights of the left and right subtree as much as poosible " yields trees whose weighted path length is bounded above by 2 + 1, 44 • H.

C.P. Schnorr improves this bound to 3 + 1, o7 • H in [8]. In the case that all internal nodes have weight o an algorithm due to T.C. Hu and A.C. Tucker [3] finds the optimum binary search tree in O (n log n) units of time and o (n) units of space. In the general case, D.E. Knuth shows how to find the optimum tree in $O(n^2)$ units of time and $O(n^2)$ units of space [5].

III. Lower Bounds :

We turn now to lower bounds. Again we will exhibit bounds which are best possible. Upper and lower bounds differ only by a constant factor; thus they define a narrow interval containing the weighted path length of the optimum (and the nearly optimal) search tree. This permits a simple a-priori test for the performance of binary search trees.

Theorem 2 :

a) ([1]) : If all internal nodes have weight zero,

(all $\beta_i = o$) then

$H \le P_{opt}$

b) Otherwise

$1/\log 3 \; H \le P_{opt}$

c) Both bounds are best possible in the following sense :

If $c_1 \sum \beta_i + c_2 \sum \alpha_j + c_3 H$ is a lower bound on the weighted path length of optimum binary search trees then $c_3 \le 1$ in case a) and $c_3 \le 1/\log 3$ otherwise. Furthermore, if $c_3 = 1$ in case a) or $c_3 = 1/\log 3$ in case b) then $c_1, c_2 \le o$.

d) Both bounds are sharp for infinitely many distributions.

Proof :

Let $\beta_o, \alpha_1, \ldots, \beta_n, \alpha_n$ be any frequency distribution and let T_{opt} be the optimum binary search tree for this distribution, let b_i (a_j) be the distance of node (i) (leaf [j]) from the root, and let $P_{opt} = \sum \beta_i (b_i + 1) + \sum \alpha_j a_j$ be the weighted path length of T_{opt}.

We define new frequency distributions. If all internal node have weight o (all $\beta_i = o$) then define

$\beta_i' = o$ for $1 \le i \le n$

$\alpha_j' = 2^{-a_j}$ for $o \le j \le n$,

otherwise define

$\beta_i'' = 3^{-(b_i+1)}$ for $1 \le i \le n$

$\alpha_j'' = 3^{-a_j}$ for $o \le j \le n$.

It is easy to see that $\sum \beta_i' + \sum \alpha_j' = \sum \beta_i'' + \sum \alpha_j'' = 1$.

The following inequality is well-known (cf. [4]). If $p_1,\ldots,p_n$ and $q_1,\ldots,q_n$ are two frequency distributions ($\sum p_i = \sum q_i = 1$) then

$$- \sum p_i \log p_i \le - \sum p_i \log q_i$$

with equality if and only if $p_i = q_i$ for all i.

It yields in our case :

$$\begin{aligned} H &= \sum \alpha_j \log 1/\alpha_j \\ &\le \sum \alpha_j \log 1/\alpha_j' \\ &\le \sum \alpha_j \log 2^{a_j} = P_{opt} \end{aligned}$$

if all internal nodes have weight zero and

$$\begin{aligned} H &= \sum \beta_i \log 1/\beta_i + \sum \alpha_j \log 1/\alpha_j \\ &\le \sum \beta_i \log 1/\beta_i'' + \sum \alpha_j \log 1/\alpha_j'' \\ &\le \sum \beta_i \log 3^{(b_i+1)} + \sum \alpha_j \log 3^{a_j} \\ &\le (\log 3)\ P_{opt} \end{aligned}$$

otherwise with equality if $\beta_i = \beta_i'$ (β_i'') and $\alpha_j = \alpha_j'$ (α_j'') for all i and j.

Assume now that $\beta_i = \beta_i'$ (β_i'') and $\alpha_j = \alpha_j'$ (α_j'') for all i and j. Then it is easy to see that the approximation algorithm of section II constructs T_{opt}. Thus $P_{opt} = H$ if all internal nodes have weight o and $P_{opt} = 1/\log 3\ H$ otherwise. The lower bounds stated above are hence sharp for infinitely many distributions.

Part c) of the theorem is now inferred easily. The details are left to the reader.

q.e.d.

IV. Conclusion

We proved that P_{opt}, the weighted path length of the optimum binary search tree, lies in the following interval

$1/\log 3\ H \le P_{opt} \le 1 + H$ if all leaves have weight o

$H \le P_{opt} \le 2 + H$ if all internal nodes have weight o

$1/\log 3\ H \le P_{opt} \le 1 + \sum \alpha_j + H$ otherwise.

All bounds are best possible. Furthermore, we exhibited an approximation algorithm which constructs trees, whose path legth lies in the intervals stated above, and which can be implemented to work in O (n log) units of time and O (n) units of space.

Acknowledgement :

I want to thank Prof. C.P. Schnorr for many extremely stimulating discussions on the subject of this paper.

Bibliography :

1. E.N. Gilbert and E.F. Moore : Bell System Techn. Journal 38 (1959), 933 - 968.
2. T.C. Hu and K.C. Tan : Least Upper Bound on the Cost of Optimum Binary Search Trees, Acta Informatica, 1, 3o7 - 31o (1972).
3. T.C. Hu and A.C. Tucker : Optimal Computer Search Trees and variable length alphabetic codes, Siam J. Applied Math. 21, 514 - 532, (1971).
4. T. Kameda and K. Weihrauch : Einführung in die Kodierungstheorie, B I Skripten zur Informatik, Vol. 7.
5. D.E. Knuth : Optimum Binary Search Trees, Acta Informatica, 1, 14 - 25, 1971.
6. D.E. Knuth : The Art of Computer Programming, Vol. 3.
7. K. Mehlhorn : Nearly Optimum Binary Search Trees, Preprint, Fachbereich 1o, Universität des Saarlandes, 1974.
8. C.P. Schnorr : Two Algorithms for Nearly Optimal Binary Search Trees, Preprint, Fachbereich Mathematik, Universität Frankfurt, 1974.