OPTIMAL ALPHABETIC SEARCH TREES WITH RESTRICTED MAXIMAL HEIGHT

Russell L. WESSNER
969 W. Main Road, Apt. 1603, Middletown, R.I. 02840, USA

Received 4 August 1975, revised manuscript received 8 October 1975

Algorithms, probabilistic searching

Knuth [3] has shown that an optimal alphabetic binary search tree for \(2n+1\) weights can be constructed in time \(O(n^2)\). In this paper we will look at the problem of constructing an optimal alphabetic binary search tree whose height is less than or equal to a given bound \(L\). In [1], Garey presents an algorithm which runs in time \(O(Ln^2)\) for the special case where \(n\) records are stored at external nodes and the records do not have to be in alphabetic order. The algorithm in this paper also runs in time \(O(Ln^2)\) and solves the more general case where records are stored in internal nodes and are in alphabetic order and the external nodes represent unsuccessful searches.

The terminology used in this paper is essentially the same used by Knuth [2, pp. 434–435]. We are given \(2n+1\) weights \(p_1, p_2, ..., p_n\) and \(q_0, q_1, ..., q_n\), where

\[
m = p_1 + p_2 + ... + p_n + q_0 + q_1 + ... + q_n.
\]

\[
p_i/m = \text{probability that } k_i \text{ is the search argument},
\]

\[
q_i/m = \text{probability that the search argument lies between } k_i \text{ and } k_{i+1},
\]

(by convention, \(q_0/m\) is the probability that the search argument is less than \(k_1\), and \(q_n/m\) is the probability that the search argument is greater than \(k_n\)). The cost of a binary tree with these weights is defined to be

\[
\sum_{j=1}^{h} p_j \left( \text{level} \left( \frac{j}{2} \right) + 1 \right) + \sum_{k=0}^{n} q_k \left( \text{level} \left( \frac{k}{2} \right) \right),
\]

where \(\frac{j}{2}\) is the \(j\)th internal node in symmetric order, \(\frac{k}{2}\) is the \((k+1)\)st external node, and the root has level zero. The problem is to construct the binary tree for this set of weights which has a height of less than or equal to \(L\) and which has minimal cost among all such trees. We shall call such a tree an optimal \(L\)-restricted alphabetic binary search tree for \((p_1, p_2, ..., p_n; q_0, q_1, ..., q_n)\).

If \(T\) is an optimal \(L\)-restricted alphabetic binary search tree for \((p_1, p_2, ..., p_n; q_0, q_1, ..., q_n)\) with root node \(\frac{k}{2}\) then the left subtree is an optimal \((L-1)\)-restricted alphabetic binary search tree for \((p_1, p_2, ..., p_{k-1}; q_0, ..., q_{k-1})\) and the right subtree is optimal \((L-1)\)-restricted alphabetic binary search tree for \((p_{k+1}, p_n; q_k, q_{k+1}, ..., q_n)\) Any improvement in a subtree leads to an improvement in the whole tree. Using this principle we can systematically find larger and larger optimum subtrees.

Let \(c(i, j, v)\) be the cost of an optimal \(v\)-restricted alphabetic binary search tree for \((p_{i+1}, p_{i+2}, ..., p_j; q_i, q_{i+1}, ..., q_j)\). By convention if no \(v\)-restricted alphabetic binary search tree exists, i.e., \(v < \log_n (j-i+1)\), then \(c(i, j, v) = \infty\).

Let

\[
w(i, j) = p_{i+1} + p_{i+2} + ... + p_j + q_i + q_{i+1} + ... + q_j,
\]
\(c(i, j, v)\) and \(w(i, j)\) are defined for \(0 \leq i \leq j \leq n, 1 \leq v \leq L\). It follows that

\[
c(i, i, v) = 0
\]

\[
c(i, j, v) = w(i, j) + \min_{i < k \leq j} (c(i, k-1, v-1) + c(k, j, v-1))
\]

since the minimum possible cost of a tree with root \(r\) whose height is less than or equal to \(v\) is \(w(i, j) + c(i, k-1, v-1) + c(k, j, v-1)\). When \(i < j\) let \(R(i, j, v)\) be the set of all \(k\) for which the minimum is achieved in eq. (1); this set specifies all the possible roots of the optimum tree. Using eq. (1), we can construct an optimal \(L\)-restricted alphabetic search tree in time \(O(Ln^3)\) using methods similar to that used in [3]. This paper will show that this basic algorithm can be improved using the same technique Knuth used in [3].

**Lemma.** Let \(\Delta(i, j, v) = c(i, j, v) - c(i, j-1, v)\). Then \(\Delta(i, j, v) \geq \Delta(i+1, j, v)\), for \(j \geq i + 2\).

**Proof.** The proof is by induction on \(j - i\). For \(j - i = 2\), \(c(i, i+1, v) = q_i + p_{i+1} + q_{i+1}\):

\[
c(i, i+2, v) = \min \left\{ (q_i + p_{i+1} + 2q_{i+1} - 2p_{i+2} + 2q_{i+2}, (2q_i + 2p_{i+1} + 2q_{i+1} + p_{i+2} + q_{i+2}) \right\}
\]

\[
\Delta(i, i+2, v) = \min \left\{ (q_{i+1} + 2p_{i+2} + 2q_{i+2}, (q_i + p_{i+1} + q_{i+1} + p_{i+2} + q_{i+2}) \right\}
\]

It follows that \(\Delta(i, i+2, v) \geq \Delta(i+1, i+2, v)\), since \(\Delta(i+1, i+2, v) = q_{i+1} + p_{i+2} + q_{i+2}\).

Assume that the lemma is true for \(j - i < n\); then we have to prove it is true for \(j - i = n\). If \(v < \log_2(n+1)\) then the lemma is trivially true. Assume \(v \geq \log_2(n+1)\).

Let \(k_1 \in R(i, i+n-1, v), k_2 \in R(i+1, i+n-1, v), k_3 \in R(i, i+n, v), k_4 \in R(i+1, i+n, v)\). These \(k\)'s exist since \(v \geq \log_2(n+1)\).

\[
\Delta(i, i+n, v) = w(i, i+n) + c(i, k_3-1, v-1) + c(k_3, i+n, v-1)
\]

\[
- w(i, i+n-1) - c(i, k_1-1, v-1) - c(k_1, i+n-1, v-1),
\]

\[
\Delta(i+1, i+n, v) = w(i+1, i+n) + c(i+1, k_4-1, v-1) + c(k_4, i+n, v-1)
\]

\[
- w(i+1, i+n-1) - c(i+1, k_2-1, v-1) - c(k_2, i+n-1, v-1),
\]

\[
w(i, i+n) - w(i, i+n-1) = p_{i+n} + q_{i+n},
\]

\[
w(i+1, i+n) - w(i+1, i+n-1) = P_{i+n} + q_{i+n}.
\]

Therefore, we get that

\[
\Delta(i+1, i+n, v) = p_{i+n} + q_{i+n} + c(i+1, k_4-1, v-1) + c(k_4, i+n, v-1)
\]

\[
- c(i+1, k_2-1, v-1) - c(k_2, i+n-1, v-1),
\]

\[
\Delta(i, i+n, v) = p_{i+n} + q_{i+n} + c(i, k_3-1, v-1) + c(k_3, i+n, v-1) - c(i, k_1-1, v-1) - c(k_1, i+n-1, v-1).
\]

**Case 1.** \(k_2 \geq k_3\). Since \(k_1 \in R(i, i+n-1, v)\),

\[
c(i, k_3-1, v-1) + c(k_3, i+n-1, v-1) \geq c(i, k_1-1, v-1) + c(k_1, i+n-1, v-1).
\]

Therefore

\[
\Delta(i, i+n, v) \geq p_{i+n} + q_{i+n} + c(i, k_3-1, v-1) + c(k_3, i+n, v-1) - c(i, k_3-1, v-1) - c(k_3, i+n-1, v-1)
\]

\[
= p_{i+n} + q_{i+n} + c(k_3, i+n, v-1) - c(k_3, i+n-1, v-1)
\]

\[
= p_{i+n} + q_{i+n} + \Delta(k_3, i+n, v-1) = P_1.
\]
Since \( k_4 \in R(i+1, i+n, v) \)
\[
c(i+1, k_4-1, v-1) + c(k_4, i+n, v-1) \leq c(i+1, k_2-1, v-1) + c(k_2, i+n, v-1).
\]

Therefore
\[
\Delta(i+1, i+n, v) \leq p_{i+n} + q_{i+n} + c(i+1, k_2-1, v-1) + c(k_2, i+n, v-1) \\
- c(i+1, k_2-1, v-1) - c(k_2, i+n-1, v-1) \\
= p_{i+n} + q_{i+n} + c(k_2, i+n, v-1) - c(k_2, i+n-1, v-1) \\
= p_{i+n} + q_{i+n} + \Delta(k_2, i+n, v-1) = P_2.
\]

To prove the lemma for case 1 we only have to show that \( P_1 \geq P_2 \). Since \( k_2 \geq k_3 \), by repeated application of the induction hypothesis
\[
\Delta(k_3, i+n, v-1) \geq \Delta(k_2, i+n, v-1).
\]

This implies \( P_1 \geq P_2 \).

**Case 2.** \( k_2 < k_3 \). Since \( k_1 \in R(i, i+n-1, v) \),
\[
c(i, k_1-1, v-1) + c(k_1, i+n-1, v-1) \leq c(i, k_2-1, v-1) + c(k_2, i+n-1, v-1).
\]

Therefore
\[
\Delta(i, i+n, v) \geq p_{i+n} + q_{i+n} + c(i, k_3-1, v-1) + c(k_3, i+n, v-1) - c(i, k_2-1, v-1) - c(k_2, i+n-1, v-1) = P_3.
\]

Since \( k_4 \in R(i+1, i+n, v) \),
\[
c(i+1, k_4-1, v-1) + c(k_4, i+n, v-1) \leq c(i+1, k_3-1, v-1) + c(k_3, i+n, v-1).
\]

Therefore
\[
\Delta(i+1, i+n, v) \leq p_{i+n} + q_{i+n} + c(i+1, k_3-1, v-1) + c(k_3, i+n, v-1) - c(i+1, k_2-1, v-1) - c(k_2, i+n-1, v-1) = P_4.
\]

To prove the lemma for case 2 and thus complete the proof of the lemma, we only have to show that \( P_3 \geq P_4 \) or equivalently that
\[
P_3 - P_4 = c(i, k_3-1, v-1) + c(k_3, i+n, v-1) - c(i, k_2-1, v-1) \\
- c(k_2, i+n-1, v-1) - c(i+1, k_3-1, v-1) - c(k_3, i+n, v-1) + c(i+1, k_2-1, v-1) + c(k_2, i+n-1, v-1) \\
= c(i, k_3-1, v-1) - c(i+1, k_3-1, v-1) - c(i+1, k_3-1, v-1) + c(i, k_2-1, v-1) - c(i+1, k_2-1, v-1).
\]

By the induction hypothesis
\[
\Delta(i, k_3-1, v-1) \geq \Delta(i+1, k_3-1, v-1),
\]
\[
c(i, k_3-1, v-1) - c(i, k_3-2, v-1) \geq c(i+1, k_3-1, v-1) - c(i+1, k_3-2, v-1).
\]

Rearranging terms we get that
\[
c(i, k_3-1, v-1) - c(i+1, k_3-1, v-1) \geq c(i, k_3-2, v-1) - c(i+1, k_3-2, v-1).
\]
By repeated application of this procedure we can get that
\[ c(i, k_3 - 1, v - 1) - c(i + 1, k_3 - 1, v - 1) \geq c(i, k_2 - 1, v - 1) - c(i + 1, k_2 - 1, v - 1) , \]
since \( k_2 < k_3 \). Therefore \( P_3 - P_4 \geq 0 \), completing the proof of the lemma.

This lemma essentially states that the increase in cost of an optimal \( v \)-restricted alphabetic binary search tree by adding \( p_{i+1}^j \) and \( q_{i+1}^j \) to the set of weights \( (p_i^j, p_{i+1}^j, \ldots, q_{i-1}^j, q_i^j, \ldots, q_j^j) \) is greater than or equal to the increase in cost by adding \( p_{j+1}^i \) and \( q_{j+1}^i \) to the set of weights \( (p_i^j, p_{i+1}^j, \ldots, p_{j-1}^j, q_{j-1}^j, q_j^j, \ldots, q_i^j) \) where \( i' > i \); by symmetry, the increase in cost by adding \( p_{i-1}^j \) and \( q_{i-2}^j \) to the set of weights \( (p_i^j, p_{i+1}^j, \ldots, p_{i-1}^j, q_{i-1}^j, q_i^j, \ldots, q_j^j) \) is greater than or equal to the increase in cost by adding \( p_{i-1}^j \) and \( q_{i-2}^j \) to the set of weights \( (p_i^j, p_{i+1}^j, \ldots, p_{j-1}^j, q_{j-1}^j, q_j^j, \ldots, q_i^j) \) where \( j' < j \).

Let \( A \) and \( B \) be non-empty sets of real numbers, then define \( A \leq B \) if the following property holds:
\[(a \in A, b \in B, \text{ and } b < a) \text{ implies } (a \in B \text{ and } b \in A).\]

**Theorem.** If \( c(i, j, v) < \infty \), then
\[ R(i, j - 1, v) \leq R(i, j, v) \leq R(i + 1, j, v) . \]

**Proof:** We will only prove that \( R(i, j - 1, v) \leq R(i, j, v) \). The proof for \( R(i, j, v) \leq R(i + 1, j, v) \) follows by symmetry.

Since \( c(i, j, v) < \infty \) there exists an optimal tree for weights \( (p_i^j, \ldots, p_{j-1}^j, q_{i-1}^j, \ldots, q_j^j) \) and an optimal tree for weights \( (p_{i+1}^j, \ldots, p_{j-1}^j, q_{i-1}^j, \ldots, q_j^j) \). Let \( s \in R(i, j, v) \) and \( r \in R(i, j - 1, v) \) such that \( s < r \). We want to prove that \( s \in R(i, j - 1, v) \) and \( r \in R(i, j, v) \). Since \( s \in R(i, j, v) \)
\[ c(i, s - 1, v - 1) + c(s, j, v - 1) \leq c(i, r - 1, v - 1) + c(r, j, v - 1) , \]
\[ c(s, j, v - 1) = c(s, j - 1, v - 1) + c(s, j, v - 1) - c(s, j - 1, v - 1) = c(s, j - 1, v - 1) + \Delta(s, j, v - 1) , \]
\[ c(r, j, v - 1) = c(r, j - 1, v - 1) + c(r, j, v - 1) - c(r, j - 1, v - 1) = c(r, j - 1, v - 1) + \Delta(r, j, v - 1) . \]
Eq. (2) can be rewritten as
\[ c(i, s - 1, v - 1) + c(s, j - 1, v - 1) + \Delta(s, j, v - 1) \leq c(i, r - 1, v - 1) + c(r, j - 1, v - 1) + \Delta(r, j, v - 1) . \]
By repeated application of the lemma we know that
\[ \Delta(s, j, v - 1) \geq \Delta(r, j, v - 1) . \]
Therefore
\[ c(i, s - 1, v - 1) + c(s, j - 1, v - 1) \leq c(i, r - 1, v - 1) + c(r, j - 1, v - 1) . \]
But since \( r \in R(i, j - 1, v) \)
\[ c(i, r - 1, v - 1) + c(r, j - 1, v - 1) \leq c(i, s - 1, v - 1) + c(s, j - 1, v - 1) . \]
Therefore
\[ c(i, r - 1, v - 1) + c(r, j - 1, v - 1) = c(i, s - 1, v - 1) + c(s, j - 1, v - 1) , \]
and \( s \in R(i, j - 1, v - 1) \). From eqs. (3) and (5) we can infer that
\[ \Delta(s, j, v - 1) \leq \Delta(r, j, v - 1) . \]
This equation and eq. (4) proves that
\[ \Delta(s, j, v - 1) = \Delta(r, j, v - 1) . \]

and \( r \in R(i, j, v) \). This completes the proof of the theorem.

Let \( r(i, j, v) \) be a member of \( R(i, j, v) \). The result of the theorem says that we may assume

\[
  r(i, j - 1, v) \leq r(i, j, v) \leq r(i + 1, j, v),
\]

if \( v \geq \log_2(j - i + 1) \). If \( v < \log_2(j - i + 1) \) then no \( v \)-restricted alphabetic binary search tree exists and any root will be an optimum root and we can still assume

\[
  r(i, j - 1, v) < r(i, j, v) < r(i + 1, j, v).
\]

Therefore, in solving eq. (1) we only have to examine \( r(i + 1, j, v) - r(i, j - 1, v) + 1 \) values for \( k \) instead of \( j - i \) values. The total amount of work for fixed \( j - i = d \) and a fixed \( v \) is bounded by the series

\[
  \sum_{d \leq j \leq n, \atop i = j - d} (r(i + 1, j, v) - r(i, j - 1, v) + 1) = r(n - d + 1, n, v) - r(0, d - 1, v) + n - d + 1 < 2n.
\]

In solving all subproblems for fixed \( j - i = d \) and fixed \( v \) we only have to consider at most \( 2n \) tentative solutions. Since there are only \( n + 1 \) possibilities for \( d \) and \( L + 1 \) possibilities for \( v \), the total running time of the improved algorithm is only \( O(Ln^2) \).

An additional feature of this algorithm is that it constructs the optimal \( v \)-restricted alphabetic binary search tree for all \( v, 1 \leq v \leq L \), with one application of the algorithm. This provides the user with useful information about the effect on the average search time with various limits on the maximum search time.

I wish to express my appreciation to Prof. Ronald Rivest for the help he gave me in preparing this paper.

References