Binary Search Trees and Binary Code Trees

T. C. Hu M. T. Shing P. A. Tucker

September 16, 1998

A binary tree consists of n circular (internal) nodes and $n+1$ square (external) nodes, where every circular node has exactly two children and every square node has no children. There are positive weights $p_1, p_2, \ldots, p_n$ associated with the circular nodes, and positive weights $q_0, q_1, \ldots, q_n$ associated with the square nodes.

Given these $2n+1$ weights as a sequence

$$q_0, p_1, q_1, p_2, q_2, \ldots, p_n, q_n$$

the binary tree that minimizes

$$\sum_{i=1}^{n} p_i(l_i + 1) + \sum_{i=0}^{n} q_i l_i' \tag{1}$$

is called the optimum binary search tree (see [?]) where l_i and l_i' are the level numbers of the circular and square nodes, and the root is at level zero.

When all $p_i = 0$ in (??), the binary tree is called the alphabetic binary tree. Note that the weights $q_0, q_1, q_2, \ldots, q_n$ are given as a sequence, and the optimum alphabetic binary tree must have its leaves also appear in the same "left to right order," hence the name, alphabetic binary tree.

The optimum binary search tree can be constructed by dynamic programming in $O(n^2)$ time and $O(n^2)$ space, where the optimum alphabetic binary tree can be constructed in $O(n \log n)$ time and $O(n)$ space [?].

We present two new results and two open problems.

Result 1. In the usual analysis it is implicitly assumed that 3-way comparisons ($<$, $=$, $>$) are used at the circular nodes in a binary search tree. However, if 2-way comparisons are used, then it is appropriate to use the Hu-Tucker algorithm to construct the optimum alphabetic binary tree in $O(n \log n)$ time and $O(n)$ space. A slightly modified binary search algorithm can then be performed on the alphabetic binary tree.

Result 2. The classical sequence partitioning problem can be stated as follows. Given a sequence of positive numbers $a_1, a_2, \ldots, a_n$, partition it into k subsequences such that the total weight (in each subsequence) is as uniform as possible. (For example, the numbers may represent job durations, and we want to assign the jobs to processors so as to evenly distribute the load.) This problem can be solved by dynamic programming in $O(kn^2)$ time [**?**], but can be solved in $O(n \log n)$ time using the Hu-Tucker algorithm.

Open Problem 1. For many weight sequences $q_0, q_1, \ldots, q_n$, the optimum alphabetic binary tree can be constructed in linear time [**?**]. Can the optimum alphabetic tree be constructed in linear time in general?

Open Problem 2. If not, what proportion of weight sequences can be solved in linear time?