Acta Informatica 9, 23–29 (1977)

Worst-Case Analysis for Region and Partial Region Searches in Multidimensional Binary Search Trees and Balanced Quad Trees

D.T. Lee[1]* and C.K. Wong[2]**

[1] Department of Computer Science and the Coordinated Science Laboratory, University of Illinois, Urbana-Champaign, IL. 61801, USA
[2] IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598, USA

Summary. Given a file of N records each of which has k keys, the worst-case analysis for the region and partial region queries in multidimensional binary search trees and balanced quad trees are presented. It is shown that the search algorithms proposed in [1, 3] run in time $O(k \cdot N^{1-1/k})$ for region queries in both tree structures. For partial region queries with s keys specified, the search algorithms run at most in time $O(s \cdot N^{1-1/k})$ in both structures.

I. Introduction

In this paper, a file is regarded as a collection of records each of which can be defined as an ordered k-tuple $(r_0, r_1, \dots, r_{k-1})$ of values. Each component of the k-tuple is referred to as a key. Given a file, we are to perform some operations on it such as updates or retrievals. According to the specified conditions, the retrieval request, called query, to a file can be classified as follows.

A simple query, or exact match query, specifies a specific value for each key. A partial match query, on the other hand, specifies only $s<k$ key values with the remaining $k-s$ keys unspecified. A region query specifies for each key a range, i.e. an interval (l_i, u_i) for each key K_i. The counterpart, termed partial region query, specifies a range for each of the $s<k$ keys. There are many other types of queries [5] that can be posed, but here we are only concerned with region and partial region queries.

We shall consider two kinds of tree structures designed for storing information to be retrieved by "associative searches" based on composite keys. Although there exist several techniques for handling such an information retrieval system [4], these two recently developed data structures, namely quad trees [3] and k-d trees [1], deserve some more investigation.

* This author's research was supported in part by the National Science Foundation under grant MCS-76-17321 and in part by the Joint Service Electronics Program under contract DAAB-07-72-C-0259

** To whom offprint requests should be sent

A k-dimensional quad tree is a multiway tree in which every node has 2^k children each corresponding to a possible outcome of the comparison of two records. A *k-d* tree is a binary tree in which every node has 2 children each corresponding to an outcome of the comparison of two records based on a certain key chosen as a "discriminator". Without loss of generality, Bentley [1] defined the following selection rule for the discriminator of each node. All nodes at level i have key $K_{i \,(\mathrm{mod}\, k)}$ as discriminators; the root is assumed to be at level 0.

For both data structures it has been shown that the average and worst-case running times for an exact match query are $O(\log N)$, where N is the total number of records. For *k-d* tree, the running time for a partial match query in the worst case has also been shown to be $O(N^{1-s/k})$ where s is the number of keys specified in the partial match query. Rivest also conjectured in [5] that the average time for a partial match query must be lower bounded by $O(N^{1-s/k})$. In [2], Bentley and Stanat showed that the expected time for the region query for 2-dimensional quad tree is $xyN+(x+y)(3N)^{1/2}+\log_4(3N)$, where x, y are the edge sizes of the region, and $0 \leqq x,\ y \leqq 1$, based on the assumption that all the records are within a unit square and are "perfectly" distributed, i.e. the root is located at the center of the square, the four nodes at level 1 are located respectively at the centers of the corresponding quadrants, etc. Other than this, currently there is no known analysis for the region query for these two data structures. Our objective here is to present a worst-case analysis for region and partial region queries for *k-d* trees and balanced quad trees. (Note that balanced quad trees are not always achievable).

II. Analysis of Region Query in *k-d* Trees

In this section we shall analyze the worst-case running time of the REGION-SEARCH algorithm[1] described in [1] using *k-d* tree as the data structure. To illustrate the approach to the worst-case analysis of the algorithm, it is sufficient to discuss the two-dimensional case ($k=2$) in which all records have 2 keys. We can represent the file in the plane with each record being a point whose x- and y-coordinates correspond to key values. Without loss of generality, we shall assume that all the records lie in the first quadrant and the total number of records is $N=2^m-1$ for some positive integer m. Thus the "ideal" 2-*d* tree with 2^m-1 nodes is a complete binary tree with all leaves appearing on level m. (It has been shown in [1] that such a tree is always attainable.) The region is rectilinearly oriented (or equivalently a rectangle) and can be represented by an array $L(1)$, $U(1)$, $L(2)$, $U(2)$ where $L(1)$ and $U(1)$, $L(2)$ and $U(2)$ are lower and upper bounds for each key respectively. As will be explained later, without affecting the order-of-magnitude complexity, we assume that the region has two coordinate axes as two lower bounds, i.e. $L(1)=L(2)=0$ (see Fig. 1). We shall use the number of node visits made in the course of searching down the tree as a measure for the running time. Since we are considering the worst possible case, for a given region we can "arrange" our records at will such that the total number of node visits is maximum. As defined

[1] Some slight modification of the algorithm is necessary. It will become clear later. Also note that our algorithm may return a "pointer" to a subtree containing desired nodes in a single counted access, while the algorithms in [1–3] count an access for each desired node