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 $(r_0, r_1, \ldots, r_{k-1})$ 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 $K_{i \pmod k}$ 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

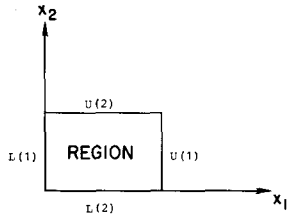

Fig. 1

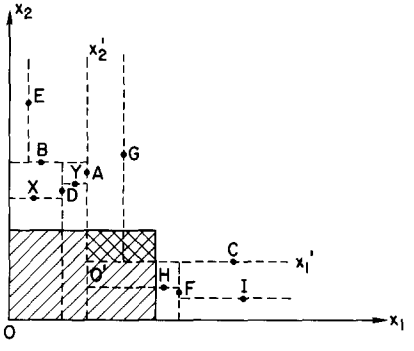

Fig. 2

in [1], associated with each node of the *k-d* tree there is a "bounds array" which specifies a region in which all its descendants must lie. In other words, each node represents a subfile. The root represents the entire file.

We use the REGIONSEARCH algorithm and make a node visit if the associated region *overlaps* the specified region except when the associated region lies completely *within* the specified region, in which case we do not search the subfile but retrieve the entire subfile. (In the original algorithm as described in [1], a node visit is made even in this case.) It is conceivable that the case in which no records, after the search, are found in the specified region may occur. We shall show that the total number of node visits in the worst case is $O(k \cdot N^{1-1/k})$.

In Figure 2, the root of the 2-*d* tree is A, and its two children are B and C. Thus, A represents the entire file, B and C represent two subfiles. Since we are considering the worst case, we can assume that the two subfiles represented by B and C overlap the specified region. Therefore we have made a node visit to A, a node visit to B and a node visit to C. At this point, one of the subfiles of B is either completely outside the region (e.g. the subfile represented by E as shown in Fig. 2) or completely within the region (e.g. the subfile represented by D if B were in the region). We can eliminate it from further consideration. Thus, among the four subfiles represented by D, E, F, and G at least one can be eliminated. Similarly, the subfile represented by I at level 3 can also be eliminated (see Fig. 3). Note that the subfile represented by G can also be "pruned" in the same way as the file represented by A since if we move the coordinate axes x_1 and x_2, respectively, to x_1' and x_2' as shown in Figure 2 such that x_1'-axis passes through C and x_2'-axis passes through A, we have a similar configuration as before. (This also justifies our previous assumption that the specified region has two coordinate axes as two lower bounds.) What happens on the subtrees represented by B and F are exactly the same except F is at one lower level. That is, from node B we just follow one branch down the tree to D, then we trace both subtrees X and Y, and follow one branch of nodes X and Y respectively, and so on. We can represent the tree pictorially as in Figures 4 and 5, where the square nodes represent subtrees (or subfiles) which need not be searched. In Figure 4, the subfile represented by G is similar to that represented by A except that G is at two levels lower and the subfile represented by H is similar to that by D. Figure 5 shows the tree configuration for the subfile represented by D.

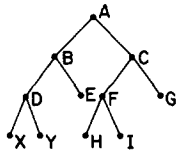

Fig. 3

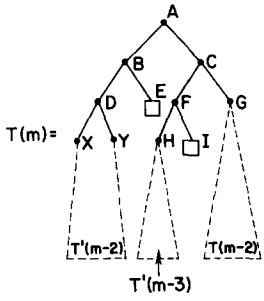

Fig. 4

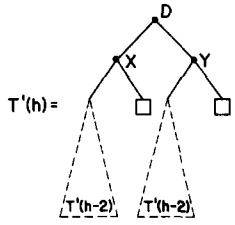

Fig. 5

Let $T'(h)$ denote the number of node visits for the subfile represented by D, where h is the height of the tree and let $T(m)$ be defined similarly for the entire file.

Thus, we have the following recurrence relations:

$$T(m)=1+1+1+T(m-2)+T'(m-2)+1+T'(m-3) \tag{1}$$

$$T'(h)=1+1+1+2T'(h-2) \tag{2}$$

with initial conditions $T'(0)=T(0)=0$, $T'(1)=T(1)=1$. Note that on the right-hand side of (1), the terms correspond to nodes A, B, C, G, D, F, and H. The terms of (2) correspond to D, X, Y and their subtrees.

From (2), we have

$$T'(h)=\begin{cases}4\cdot 2^i-3, & \text{for } h=2i+1,\\ 3\cdot 2^i-3, & \text{for } h=2i.\end{cases}$$

After substitution in (1) and solving for $T(m)$, we have

$$T(m)=\begin{cases}5(2^{m/2}-1)-m, & \text{for } m \text{ even}, \quad (3)\\ 7(2^{\lfloor m/2\rfloor}-1)-m+2, & \text{for } m \text{ odd}. \quad (4)\end{cases}$$

Since $N=2^m-1$, we can rewrite $T(m)$ as

$$T(m)=O(2\cdot N^{1/2}),$$

i.e. $O(k\cdot N^{1-1/k})$ for $k=2$.

For the k-dimensional case, it becomes a matter of counting the number of "bounded" regions. Notice that whenever we encounter a bounded region we can ignore one of the subtrees. For example, after we made a node visit to B, we have a bounded region associated with node D (see Fig. 2). If this region overlaps but is not within the specified region, the subfile represented by E can be eliminated. If it is within the specified region, the subfile represented by D can be retrieved. In the worst case we shall assume that the latter does not happen. Based on this observation, we can evaluate the total number of node visits in terms of the number of bounded regions which overlap the specified region. To be precise, the total number of node visits is equal to the total number of "unbounded" regions caused

by the partition of each node. In the following discussion, the total number of nodes is assumed to be $N=2^{mk}-1$.

For simplicity, we shall follow the discriminator selection rule defined by Bentley [1]. Consider the first quadrant in the k-dimensional space with coordinates $x_1, x_2, \ldots, x_k$. We shall use hyperplanes perpendicular to the x_i-axis to partition the existing regions. The coordinate hyperplanes also serve as region boundaries. At level 0 we have the entire quadrant, hence the number of unbounded region is 1. At level 1, we draw a hyperplane perpendicular to the x_1-axis. This is the first visit to the x_1-axis. Then we do the same for $x_2, \ldots, x_k$. On the second visit to the x_1-axis, we draw two hyperplanes perpendicular to the x_1-axis, one in each of the two regions generated on the first visit. Thus the number of resulting regions is 4. Do the same for $x_2, \ldots, x_k$, and so on. At level $i<k$, the regions formed are unbounded. At level k, the first bounded region occurs. In general, at level p, the number of visits to the x_i-axis is $f_p(x_i)=1+\left\lfloor\frac{p-i}{k}\right\rfloor$. To see this, let $p=l_1\ k+l_2$, for some nonnegative integers l_1 and l_2. If $l_2 \geqq i$, then $f_p(x_i)=l_1+1$, otherwise $f_p(x_i)=l_1$. Thus,

$$f_p(x_i)=1+\left\lfloor\frac{p-i}{k}\right\rfloor.$$

Therefore, the number of regions generated by these visits is $2^{1+\lfloor\frac{p-i}{k}\rfloor}$. All of them except one are bounded by two hyperplanes perpendicular to the x_i-axis. Thus, the total number of bounded regions at level p is given by

$$B(p)=\prod_{i=1}^{k}(2^{1+\lfloor\frac{p-i}{k}\rfloor}-1).$$

The total number of unbounded regions at level p is therefore

$$S(p)=2^p-B(p)$$

and the total number of node visits is

$$\begin{aligned} V_k(mk, N) &= \sum_{p=0}^{mk-1} S(p) \\ &= \sum_{p=0}^{mk-1}\left[2^p-\prod_{i=1}^{k}(2^{1+\lfloor\frac{p-i}{k}\rfloor}-1)\right]. \end{aligned} \tag{5}$$

After expanding and rearranging terms, we have

$$\begin{aligned} V_k(mk, N) &= 2^{mk}-1-\sum_{i=1}^{m-1}(2^i-1)^k-\sum_{t=1}^{m-1}\sum_{i=1}^{k-1}(2^t-1)^i(2^{t+1}-1)^{k-i} \\ &= 2^{mk}-1-\sum_{i=1}^{m-1}(2^i-1)^k-\sum_{t=1}^{m-1}\frac{(2^{t+1}-1)^k(2^t-1)-(2^t-1)^k(2^{t+1}-1)}{2^t} \\ &= (2^{mk}-1)-(2^m-1)^k+A+\frac{1}{2^m}(2^m-1)^k, \end{aligned} \tag{6}$$

where $A=\sum_{t=1}^{m}\frac{(2^t-1)^k}{2^t}$. Using binomial expansion for the numerators of terms in A and noting that in (6) the term 2^{mk} is cancelled, we have

$$\begin{aligned} V_k(mk, N) &\approx \left[(k+2)+\frac{1}{2^{k-1}-1}\right] 2^{m(k-1)} \\ &= O(k \cdot 2^{mk-m}) \\ &= O(k \cdot N^{1-1/k}). \end{aligned} \tag{7}$$

Also note that for $k=1$, $V_1(m, N)=m=O(\log N)$ and for $k=2$, (6) reduces to (3).

III. Analysis of Region Query in Balanced Quad Trees

We can apply the same technique to get the worst-case performance of region query in the k-dimensional balanced quad trees. Assume the number of levels in the quad tree is m and each node has 2^k children. Thus, the total number of nodes in a balanced quad tree becomes $N_1=\frac{2^{mk}-1}{2^k-1}$. The number of bounded regions at level p is given by $B(p)=(2^p-1)^k$. Therefore, the total number of node visits, i.e. the total number of unbounded regions, is given by

$$\begin{aligned} V_q(m, N_1) &= \sum_{p=0}^{m-1} 2^{pk}-(2^p-1)^k \\ &= O(k \cdot N_1^{1-1/k}), \end{aligned} \tag{8}$$

which is of the same form as (7).

IV. Analysis of Partial Region Query

In the preceding sections, we considered the case where the region was a "closed" hyper-rectangle, i.e. a closed region in which every key had a specified interval. Here we shall deal with a variant of the region query called partial region query in which only $s<k$ keys have specified intervals.

We have a somewhat similar situation as before except for the following: If we are at a node whose i^{th} key is chosen as the discriminator, and if the corresponding partial region query does not have a specified interval for the key, then we have to make a node visit to each subtree of the node because in that particular direction we have an "infinite" region and the two regions associated with the two subtrees of the node will overlap the "infinite" region. As in the analysis for the partial match query in [1], the pessimal arrangement will be that the specified s keys are the last s keys. We shall have a geometric explosion before we can prune the search tree. For k-d trees, we have the following formula for the total number of node visits:

$$V_k'(mk, N)=\sum_{p=0}^{mk-1}\left[2^p-\prod_{i=1}^{k-s} 2^{1+\lfloor\frac{p-i}{k}\rfloor}\cdot\prod_{i=k-s+1}^{k}(2^{1+\lfloor\frac{p-i}{k}\rfloor}-1)\right]. \tag{9}$$

For balanced quad trees, we have:

$$V'_q(m, N_1) = \sum_{p=0}^{m-1} [2^{pk} - 2^{p(k-s)}(2^p - 1)^s]. \tag{10}$$

Expanding and rearranging terms in (9), we have

$$V'_k(mk, N) = (2^{mk} - 1) - (A + B),$$

where $A = (2^{t+1} - 1) \sum_{j=1}^{m-1} (2^j - 1)^s \, 2^{jt}$ and

$$B = \sum_{j=1}^{m-1} 2^{(j+1)t} [2^{(j+1)} - 1]^s - 2 \sum_{j=1}^{m-1} 2^{(j+1)t} (2^j - 1)^s + \sum_{j=1}^{m-1} \frac{2^{(j+1)t}(2^j - 1)^s - 2^{(j+1)t}(2^{j+1} - 1)^s}{2^j}.$$

Thus, $V'_k(mk, N) \approx (s+2)\, 2^{mk-m} = O(s \cdot N^{1-1/k})$.

From (10), $V'_q(m, N_1)$ can be easily verified to be $O(s \cdot N_1^{1-1/k})$.

V. Conclusions

As one may have noticed in (5) that the same number of node visits would have been made if we permuted the order in which the discrimator key was selected, since Equation (5) is symmetric with respect to all keys K_i. In other words, in every consecutive k levels the order in which the index of the discriminator key is selected can be any permutation of (1, 2, ..., k). This gives us the flexibility of selecting the discriminator keys to facilitate some applications of the *k-d* trees without affecting the running time of the region queries.

The average-case analysis for the region queries or partial region queries in these two tree structures is still an open problem.

Acknowledgments. The authors are grateful to Prof. J.L. Bentley of the University of North Carolina at Chapel Hill for many valuable comments on the first draft of this paper.

References

1. Bentley, J.L.: Multidimensional binary search trees used for associative searching. Comm. ACM **18**, 509–517 (1975)
2. Bentley, J.L., Stanat, D.F.: Analysis of range searches in quad trees. Information Processing Lett. **3**, 170–173 (1975)
3. Finkel, R.A., Bentley, J.L.: Quad trees: a data structure for retrieval on composite key. Acta Informat. **4**, 1–9 (1974)
4. Knuth, D.E.: The art of computer programming, Vol. 3. New York: Addison-Wesley 1973
5. Rivest, R.L.: Partial-match retrieval algorithms. SIAM J. Comput. **5**, 19–50 (1976)

Received June 16, 1976