RAIRO. INFORMATIQUE THÉORIQUE ET APPLICATIONS

M. TALAMO
G. GAMBOSI

An application of *m*-ary trees to the design of data structures for geometric searching problems

RAIRO. Informatique théorique et applications, tome 23, n° 2 (1989), p. 165-176

<http://www.numdam.org/item?id=ITA_1989__23_2_165_0>

Informatique théorique et Applications/Theoretical Informatics and Applications
(vol. 23, n° 2, 1989, p. 165 à 176)

AN APPLICATION OF m-ARY TREES TO THE DESIGN OF DATA STRUCTURES FOR GEOMETRIC SEARCHING PROBLEMS (*)

by M. TALAMO (1) and G. GAMBOSI (1)

Communicated by G. AUSIELLO

Abstract. – *An efficient solution to the ECDF searching problem is presented which extends to ECDF counting.*

Such solution relies on the use of m-ary trees combined with the introduction of particular cumulated sets and makes it possible to obtain parametric bounds which allow to "tune" the data structures introduced.

Résumé. – *Dans cet article on présente une solution efficiente au problème de la recherche ECDF, que l'on peut appliquer aussi au problème général ECDF.*

La solution est basée sur l'usage d'arbres m-aires et d'ensembles cumulatifs, et nous permet d'établir des limites paramétriques pour la complexité des algorithms proposés.

1. INTRODUCTION

Searching or counting all points included in some interval in a d-dimensional space is one of the fundamental problems in Computer Science and, in particular, in the fast growing field of Computational Geometry. Such problems are differently denoted depending on the type of intervals considered. In particular:

– The ECDF (searching-counting) problem [8] regards intervals open at left, *i.e.*, given a set of points S and a point $x=(x_1, x_2, \ldots, x_d)$, it asks for all points $z=(z_1, z_2, \ldots, z_d)$ in S such that $z_i \leqq x_i$ $(i=1, \ldots, d)$.

– The Orthant (searching-counting) problem [15] concerns intervals open either at left or at right, *i.e.*, given a set of points S and a point

(*) Received December 1986, revised May 1987.
(1) Istituto di Analisi dei Sistemi ed Informatica del CNR, Viale Manzoni 30, 00185 Roma, Italy.

Informatique théorique et Applications/Theoretical Informatics and Applications
0296-1598 89/02 165 12/$3.20/

$x=(x_1, x_2, \ldots, x_d)$, it asks for all points $z=(z_1, z_2, \ldots, z_d)$ in S such that, for each $i=1, \ldots, d$, either $z_i \leqq x_i$ or $z_i \geqq x_i$.

– The (Orthogonal) Range (searching-counting) problem [5] treats the case of closed intervals, *i.e.* given a set of points S and two points $x=(x_1, x_2, \ldots, x_d)$ and $y=(y_1, y_2, \ldots, y_d)$, it asks for all points $z=(z_1, z_2, \ldots, z_d)$ in S such that $x_i \leqq z_i \leqq y_i$ $(i=1, \ldots, d)$.

Many efforts have been devoted to the design of data structures for set S and of search algorithms on such structures which allow an efficient treatment of such queries, and a large amount of papers which are concerned with the definition of upper and lower bounds for such problems is available in the literature. The ECDF searching/counting problems were introduced in [8] in the framework of statistics, while [1] gives a recursive solution to the problem as an example of the general paradigm of multidimensional divide and conquer. A solution for the case $d=3$ has been developed in [11] by reduction to a paper stabbing problem.

In general, however, the ECDF problems have been studied in the framework of the more general (orthogonal) range searching/counting problems. Such problems have been extensively studied for their relevance, mainly in the context of database physical organization [2, 3, 12], leading to the introduction of quad trees and multidimensional binary $(k-d)$ trees [16].

Successively, the concept of decomposable searching problem, introduced in [4], applied to range searching has led to the design of more efficient data structures [6, 26] and to the introduction of dynamization capabilities [19, 7, 22, 20, 28]. In particular, Willard [26] obtained data structures with query time $Q(n)=O(\lg^{d-1} n)$, space $S(n)=O(n \lg^{d-1} n)$ and preprocessing time $P(n)=O(n \lg^{d-1} n)$.

Recently, Chazelle [10] applied the approach of filtering search to range searching obtaining a saving of $\lg \lg n$ on the space bound.

Lower bounds for orthogonal range searching queries are discussed in [13, 14, 27] under different computation models, while in [17, 23, 9] some relations between range searching and other geometric problems are presented.

The basic idea of this paper is to introduce a second type of divide and conquer, together with the one drived from the usual techniques of partitioning the interval considered allowing the representation of overlapped regions. Such a new approach is defined by introducing for each interval R a sequence

of b sub-intervals $R_1, \ldots, R_b$ such that:

1. $R_i \subset R$ and $R_i \subset R_{i+1}$ for each $i = 1, \ldots, b-1$.
2. $R_b = R$.

It is easy to realize that this decomposition is powerful enough to express the interval partitioning typical of divide and conquer: in fact, denoting as low (R) [up (R) respectively] the lower (upper) of interval R, such a partition is defined by [low(R_1), up(R_1)], [up(R_1), up(R_2)], ..., [up(R_{b-1}), up(R_b)]. Such solution will make it possible, for a given i, to access in a single step all points less than up (R_i) and, at the same time, to (eventually) decompose interval [up(R_i), up(R_{i+1})] ([low(R_1), *up*(R_1)] for the lowest sub-interval). Hence, only one interval will be considered for each level of decomposition in an ECDF query, thus obtaining access to $\lg_b n$ nodes.

In Paragraph 2 the basic solution to the ECDF searching/counting problem is presented: such a solution is based on the approach introduced above and on an extension of the concept of "layered tree" defined in [26] to m-ary trees. A data structure is introduced which solves the ECDF searching (counting) problems with query time $Q(n) = O(\lg_b^{d-1} n + k)$, where k is the dimension of the output, $(O(\lg_b^{d-1} n))$, space $S(n) = O(n(b \lg_b n)^{d-1})$ and pre-processing time $P(n) = O(n(b \lg_b n)^{d-1})$, where n is the number of points considered and b the degree of the partition introduced.

In particular, for $b = \lg n / \lg \lg n$ such bounds become:

- $Q(n) = O((\lg n / \lg \lg n)^{d-1})$,
- $S(n) = O(n(\lg n / \lg \lg n)^{2d-2})$,
- $P(n) = O(n(\lg n / \lg \lg n)^{2d-2})$.

respectively, thus improving the results of [26] with respect to query time. In general, however, ranging b in $[1, \ldots, |S|]$ makes it possible to obtain data structures ranging from a matrix representation to a pure range tree.

In Paragraph 3 some considerations on possible extensions and related works are presented.

2. THE DATA STRUCTURE

The solution presented here relies on the recursive decomposition of sets of points belonging to R^d into disjoint subsets of (approximately) the same size according to their x_1-coordinate: hence, given a (generic) set $S \subseteq R^d$ of $|S|$ points in a d-dimensional space which are ordered according to their x_1-coordinate, such a set is decomposed in b subsets (corresponding to intervals

on the x_1 axis) where

$$|S_1|=|S_2|=\ldots=|S_{b-1}|=\frac{|S|}{b} \quad \text{and} \quad |S_b|=|S|-(b-1)\frac{|S|}{b}.$$

Furthermore, for each set S a (cumulative) function f: $[1, \ldots, b] \to 2^S$ is defined such that $f(i)=\bigcup_{j \leqq i} S_j$. It is easy to realize that

$$|f(i)|=i\frac{|S|}{b} \quad \text{for} \quad i=1, \ldots, b-1 \quad \text{and} \quad f(b)=S.$$

Hence, each set S_i will contain all points in S contained in a particular interval $[a_i, a_{i+1}]$ on the x_1-axis (where $a_1=-\infty$ and $a_{b+1}=+\infty$), while each set $f(i)$ includes all points in the interval $[a_1, a_{i+1}]$. In figure 1 an example of such a decomposition is presented for a case with $d=2$, $b=3$, $|S|=10$.

Such a decomposition can be represented by a b-ary tree T such that each node represents a subset of S (in particular an interval on the total ordering given by the x_1-coordinate). In the following, we will denote as:

- $S(n)$ the set of points represented at node n;
- $s_i(n)$ the i-th son of node n in T.
- $f_i(n)$ the value $f(i)$ of the function f relative to set $S(n)$;
- $x_i(p)$ the value of the i-th coordinate of point p.

In particular, for $d=2$, we will denote $x_1(p)$ as $x(p)$ and $x_2(p)$ as $y(p)$.

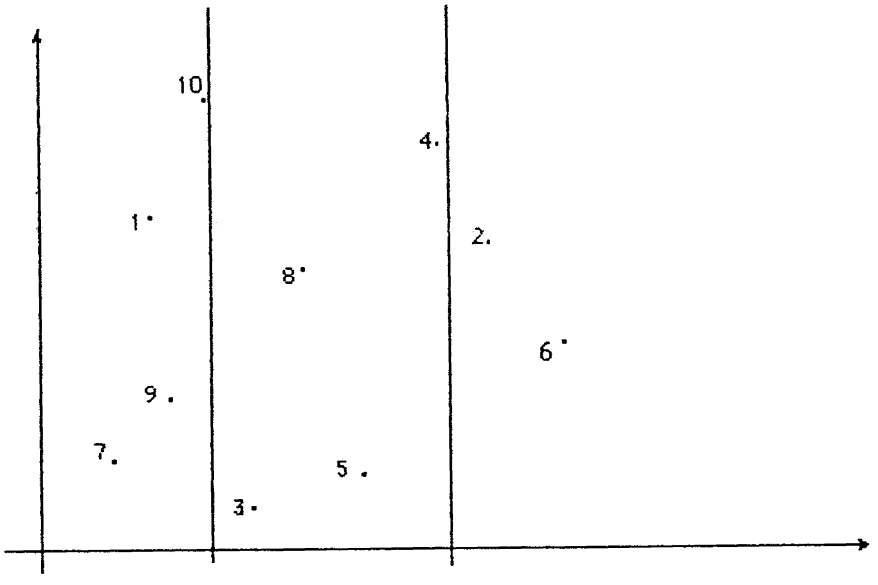
Each node n in T will have the following structure (see *fig.* 2):

1. A set of b pointers $s_1, \ldots, s_b$ to the sons $s_1(n), \ldots, s_b(n)$ of n. Pointer s_i will point to the node n' such that $S(n')=S_i$.

2. A set of b pointers $f_1, \ldots, f_b$ to the structures representing sets $f_1(n), \ldots, f_b(n)$, if such sets do exist.

In figure 3 a first example of such a structure is given for the case presented in figure 1, where pointers s_i are drawn as thick lines while pointers f_i are represented by thin lines.

Let us now turn to the organization of sets $f_i(n)$. For the sake of simplicity, let us consider the case $d=2$: in such a situation, a structure quite similar to the "layered trees" presented in [26] will be used. In such an approach, the sets $f_i(n)$ associated to a node n are implemented as arrays of points ordered according to their y-coordinate. Moreover, for each point p_j in $f_b(n)$



$S_1=\{1,7,9,10\}$ $f_1=\{1,7,9,10\}$

$S_2=\{3,4,5,8\}$ $f_2=\{1,3,4,5,7,8,9,10\}$

$S_3=\{2,6\}$ $f_3=\{1,2,3,4,5,6,7,8,9,10\}=S$

Figure 1.

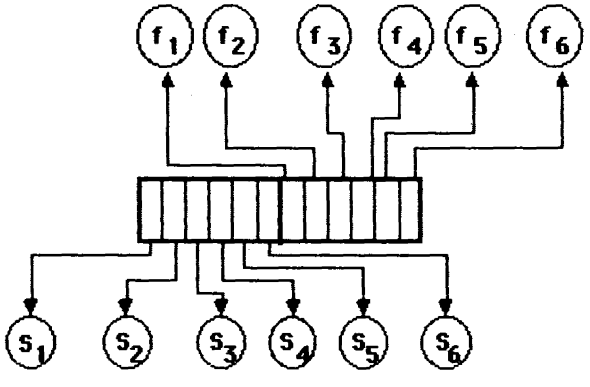


Figure 2.

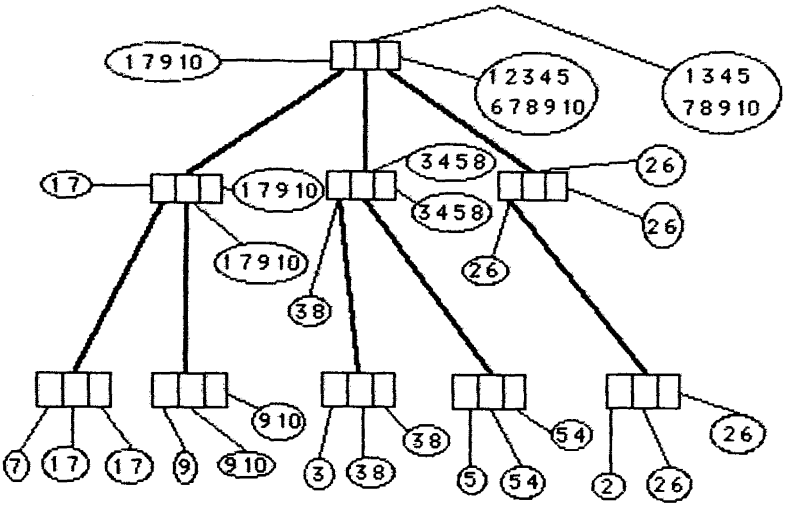


Figure 3.

$j=1, \ldots, |f_b(n)|$, there exist $2b-1$ pointers subdivided in two sets:

1. $b-1$ "brother pointers" $B_{j,n}(1), \ldots, B_{j,n}(b-1)$ such that $B_{j,n}(k)$, $k=1, \ldots, b-1$ points to the element q in $f_k(n)$ such that $y(q)$ is the greatest value of the ordinate for points in $f_k(n)$ less than or equal to $y(p_j)$.

2. b "downpointers" $D_{j,n}(1), \ldots, D_{j,n}(b)$ such that $D_{j,n}(k)$ $k=1, \ldots, b$ points to the element q' in $f_b(s_k(n))$ such that $y(q')$ is the greatest value of the ordinate for points in $f_b(s_k(n))$ less than or equal to $y(p_j)$.

These pointers will be indeed represented by the indexes of points q and q' in $f_k(n)$ and $f_b(s_k(n))$ respectively.

In figure 4 the same example of figure 3 is presented with sets $f_i(n)$ structured as above. For the sake of simplicity only pointers relative to the representation of point 5 at the first level have been drawn. Both brother pointers and downpointers have been drawn as dashed lines.

Two dictionaries dic_1 and dic_2 are moreover defined such that:

1. Given a point p, dic_1 gives, in time $O(\lg n)$, the rank $rx(p)$ of p in the ordering of points in $S \cup \{p\}$ according to their abscissa.

2. Given a point p, dic_2 gives, in time $O(\lg n)$, the rank $ry(p)$ of p in the ordering of points in $S \cup \{p\}$ according to their ordinate.

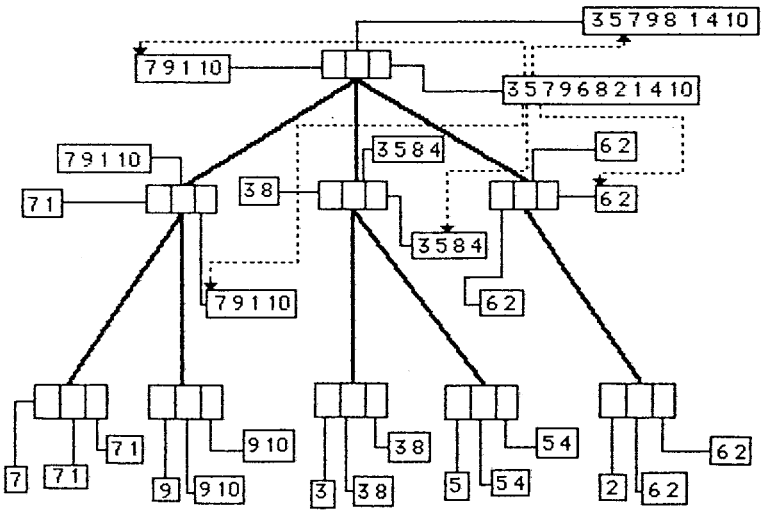


Figure 4.

Such dictionaries can be implemented by e. g. balanced trees (AVL trees, 2-3 trees, etc.).

It is now possible to state the following theorem:

THEOREM 1: *Given a set S of points in a d-dimensional space, the data structure presented above makes it possible to solve the ECDF searching problem with bound on the query time:*

- $Q(n) = O(\lg_b^{d-1} n + k)$, *where k is the dimension of the output, for* $d > 2$;
- $Q(n) = O(\lg n + k)$, *for* $d = 2$.

Proof: Let us again consider first the case $d=2$: then, the following algorithm can be designed in order to answer efficiently to ECDF searching queries.

ECDF searching algorithm
begin
  "determine on dic_1 and dic_2 the ranks $rx(p)$ and $ry(p)$";
  node-access ($rx(p)$, $ry(p)$, root, $|S|$)
end
where:
procedure node-access (*rx*, *ry*, node, *n*);
var *i*, *j*: integer;

```
begin
  i := (rx div b) + 1;
  if i > 1 then
      "output all points p_j in f_{i-1}(node) with j ≦ B_{ry, node}(i-1)";
  if "node s_i(node) exists" and (rx mod b ≠ 0) then
begin
  if i ≠ b then
    node-access (rx - n* i div b, D_{ry, node}(i), s_i(node), n div b)
  else
    node-access (rx - n* i div b, D_{ry, node}(i), s_i(node), n mod b)
  end
end;
```

For $d=2$, it is easy to realize that $Q(n)=O(\text{depth}(T)+\lg n+k)$ and, since T is a balanced b-ary tree, $Q(n)=O(\lg_b n+\lg n+k)$ with k the number of points resulting from the query: this gives a bound $O(\lg n+k)$.

In the case $d>2$ sets $f_i(n)$ can be structured as b-ary trees for dimension $d-1$: this gives the general bound $Q(n)=O(\lg_b^{d-1} n+k)$. □

THEOREM 2: *Given a set S of points in a d-dimensional space, the data structure presented above makes it possible to solve the ECDF searching problem with space $S(n)=O(n(b\lg_b n)^{d-1})$.*

Proof: For $d=2$, each point $p\in S$ occurs at most $b\lg_b n$ times in T, since it can occur in all sets $f_i(n)$ $i=1, \ldots, b$ in a node for at most all nodes in a path from root(T) to a leaf, hence in $\lg_b n$.

For $d>2$, the following recurrence holds, where $R(d)$ is the maximal number of times a point p can occur in a structure for a set S, with $|S|=n$, in d dimensions:

- $R(d)=R(d-1)\,b\lg n$;
- $R(2)=b\lg_b n$,

which gives the general expression

$$R(d)=(b\lg_b n)^{d-1} \qquad \text{and} \qquad S(n)=O(n(b\lg_b n)^{d-1}). \quad \square$$

THEOREM 3: *Given a set S of points in a d-dimensional space, the data structure presented above makes it possible to solve the ECDF searching problem with preprocessing time $P(n)=O(n(b\lg_b n)^{d-1})$.*

Proof: For $d=2$, the cost of building a single node representing m points is

$$C(m)=O\left(bm/b+\sum_{i=1}^{b-1}\frac{mi}{b}\right), \quad \text{which gives } C(m)=O(bm).$$

This results in a cost $O(bn)$ to build all nodes at a same level in T and in a global preprocessing time $P(n)=O(nb\lg_b n)$.

For $d > 2$, assume $P(n) = O(n(b \lg_b n)^{d-1})$ for dimension d. Then, the cost of building a single node representing m points in $d+1$ dimensions will be:

$$C(m) = O\left(b\,m + b\left(\frac{m}{b}(b \lg_b m)^{d-1}\right) + \sum_{i=1}^{b-1}\left(\frac{m\,i}{b}(b \lg_b m)^{d-1}\right)\right)$$

where the first term is due to the cost of deriving sets $f_i(n)$, $i = 1, \ldots, b-1$, and $S(s_i(n))$, $i = 1, \ldots, b$. The second term represents the cost of building the structures at dimension d for sets $S(s_i(n))$, while the third term accounts for the cost of building the structures at dimension d for sets $f_i(n)$.

Hence, $C(m) = O(m\,b\,(b \lg_b m)^{d-1})$ and, as a consequence,

$$P(n) = O\left(\sum_{i=1}^{b-1} b^i\, C(n/b^i)\right)$$

and $P(n) = O(n(b \lg_b n)^d)$ for a $d+1$ dimensional structure. □

COROLLARY 1: *It is possible to design a data structure which supports ECDF searching queries with bounds:*

- $Q(n) = O((\lg n / \lg \lg n)^{d-1})$;
- $S(n) = O(n(\lg n / \lg \lg n)^{2d-2})$;
- $P(n) = O(n(\lg n / \lg \lg n)^{2d-2})$.

Proof: Derives simply by substituting b with $\lg n / \lg \lg n$. □

The data structure above can be suitably modified in order to obtain an efficient solution to the ECDF counting problem. In order to do that it is possible to simply eliminate sets $f_i(n)$, $i = 1, \ldots, b-1$: the value $B_{ry,n}(i)$ will give the number of points with ordinate less than or equal to ry in the i-th sub-interval of the interval on the x axis represented at node n. Figure 5 shows such a modification for the same example of figure 4.

It is easy to derive the following theorem.

THEOREM 4: *Given a set S of points in a d-dimensional space, the data structure above makes it possible to solve the ECDF counting problem with bounds:*

- $Q(n) = O(\lg_b^{d-1} n)$;
- $S(n) = O(n(b \lg_b n)^{d-1})$;
- $P(n) = O(n(b \lg_b n)^{d-1})$.

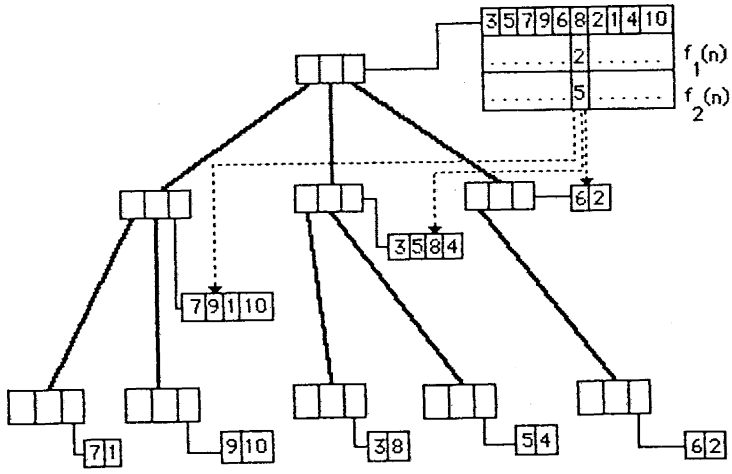


Figure 5.

Proof: Concerning query time, let us consider the following algorithm operating on the structure above:

ECDF counting algorithm
begin
$N:=0$;
"determine on dic_1 and dic_2 the ranks $rx(p)$ and $ry(p)$";
node-access ($rx(p)$, $ry(p)$, root, $|S|$, N);
"output N"
end.

where:

procedure node-access (rx, ry, node, n, N);
var i, j: integer;
begin
$i:=(rx \operatorname{div} b)+1$;
if $i > 1$ then $N=N+B_{ry,\,\text{node}}(i-1)$;
if "node s_i(node) exists" and $(rx \bmod b \neq 0)$ then
begin
if $i \neq b$ then
node-access ($rx-n^* i \operatorname{div} b$, $D_{ry,\,\text{node}}(i)$, s_i(node), $n \operatorname{div} b$, N)
else
node-access ($rx-n^* i \operatorname{div} b$, $D_{ry,\,\text{node}}(i)$, s_i(node), $n \bmod b$, N)
end
end;

Such an algorithm returns into variable N the number of points satisfying the ECDF query in a time which equals that of the ECDF searching algorithm of theorem 1 without the factor k representing the cost of enumerating all points in the output. Moreover, the space and preprocessing bounds can be easily verified to be the same as above. □

3. CONCLUSIONS

In this paper, a data structure has been presented which supports a parametrization of its structure and, then, of the related bounds on time and space. It has been shown that, for a suitable value of such a parameter it is possible to obtain quite good performances on query time.

It seems worth of further investigation the extension of such an approach to the more general orthant and orthogonal searching/counting problems; moreover, due to the introduction of m-ary trees, it seems interesting to investigate the use of such a structure on secondary memory.

REFERENCES

[1] J. L. BENTLEY, *Multidimensional Divide and Conquer*, Communications of A.C.M., vol. 23, 1980, pp. 214-229.

[2] J. L. BENTLEY, *Multidimensional Binary Search Trees Used for Associative Searching*, Communications of A.C.M., vol. 18, 1975, pp. 509-517.

[3] J. L. BENTLEY, *Multidimensional Binary Search Trees in Database Applications*, I.E.E.E. Trans. on Software Engineering, vol. 5, 1979, pp. 333-340.

[4] J. L. BENTLEY, *Decomposable Searching Problems*, Information Processing Letters, vol. 8, 1979, pp. 244-251.

[5] J. L. BENTLEY and J. H. FRIEDMAN, *Data Structures for Range Searching*, Computing Surveys, vol. 11, 1979, pp. 397-409.

[6] J. L. BENTLEY and H. A. MAURER, *Efficient Worst-case Data Structures for Range Searching*, Acta Informatica, vol. 13, 1980, pp. 155-168.

[7] J. L. BENTLEY and J. B. SAXE, *Decomposable Searching Problems #1: Static to Dynamic Transformations*, Journal of Algorithms, vol. 1, 1980, pp. 301-358.

[8] J. L. BENTLEY and M. I. SHAMOS, *A Problem in Multivariate Statistics: Algorithm, Data Structure and Applications*, Proc. 15th Annual Allerton Conf. on Communication, Control and Computing, 1977, pp. 193-201.

[9] J. L. BENTLEY and D. WOOD, *An Optimal Worst-case Algorithm for Reporting Intersection of Rectangles*, I.E.E.E. Trans. on Computers, vol. 29, 1980, pp. 571-577.

[10] B. M. CHAZELLE, *Filtering Search: a New Approach to Query Answering*, Proc. 24th I.E.E.E. Symp. on Foundations of Computer Science, 1983, pp. 122-132.

[11] B. M. CHAZELLE and H. EDELSBRUNNER, *Linear Space Data Structures for two Types of Range Search*, Tech. Report 202, Inst. of Computer Science, University of Graz, 1985.

[12] R. A. FINKEL and J. L. BENTLEY, *Quad Trees: a Data Structure for Retrieval of Composite Keys*, Acta Informatica, vol. 4, 1974, pp. 1-9.

[13] M. F. FREDMAN, *A Lower Bound on the Complexity of Orthogonal Range Queries*, Journal ACM 28, 1981, pp. 696-706.

[14] M. F. FREDMAN, *Lower Bounds on the Complexity of Some Optimal Data Structures*, SIAM Journal on Computing 10, 1981, pp. 1-10.

[15] H. N. GABOW, J. L. BENTLEY and R. E. TARJAN, *Scaling and Related Techniques for Geometry Problems*, Proc. 16th Symp. on Theory of Computing, 1984, pp. 135-143.

[16] D. T. LEE and C. K. WONG, *Worst Case Analysis for Region and Partial Region Searches in Multidimensional Binary Search Trees and Balanced Quad Trees*, Acta Informatica, vol. 9, 1977, pp. 23-29.

[17] D. T. LEE and C. K. WONG, *Finding Intersection of Rectangles by Range Search*, Journal of Algorithms, vol. 2, 1981, pp. 337-347.

[18] D. T. LEE and C. K. WONG, *Quintary Trees: a File Structure for Multidimensional Database Systems*, A.C.M. Trans. on Database Systems, vol. 5, 1980, pp. 339-353.

[19] J. VAN LEEUWEN and D. WOOD, *Dynamization of Decomposable Searching Problems*, Information Processing Letters, vol. 10, 1980, pp. 51-56.

[20] G. S. LUEKER and D. E. WILLARD, *A Data Structure for Dynamic Range Queries*, Information Processing Letters, vol. 15, 1982, pp. 209-213.

[21] J. NIEVERGELT, H. HINTERBERGER and K. SEVCIK, *The Grid File: an Adaptable, Symmetric Multikey Data Structure*, A.C.M. Trans. on Database Systems, vol. 9, 1984, pp. 38-71.

[22] M. H. OVERMARS, *The Design of Dynamic Data Structures*, Lecture Notes on Computer Science, Vol. 156, Springer Verlag, New York.

[23] M. H. OVERMARS, *The Equivalence of Rectangle Containment, Rectangle Enclosure and ECDF Searching*, Tech. Report RUU-CS-81-1, Dept. of Computer Science, University of Utrecht, 1981.

[24] J. T. ROBINSON, *The K-D-B Tree: a Search Structure for Large Multidimensional Dynamic Indexes*, Proc. of the SIGMOD Conference, 1981, pp. 10-18.

[25] J. VUILLEMIN, *A Unifying Look at Data Structures*, Communications of A.C.M., Vol. 23, 1980, pp. 229-239.

[26] D. E. WILLARD, *New Data Structures for Orthogonal Range Queries*, S.I.A.M. Journal on Computing, Vol. 14, 1985, pp. 232-253.

[27] D. E. WILLARD, *Lower Bounds for Dynamic Range Queries That Permit Subtraction* (to appear).

[28] D. E. WILLARD and G. S. LUEKER, *Adding Range Restriction Capability to Dynamic Data Structures*, Journal A.C.M., Vol. 32, 1985, pp. 597-617.