ON THE EQUIVALENCE OF B-TREES AND DETERMINISTIC SKIP LISTS

by

Michael G. Lamoureux
Bradford G. Nickerson

TR96-102, January 1996

Faculty of Computer Science
University of New Brunswick
Fredericton, N.B. E3B 5A3
Canada

Phone: (506) 453-4566
Fax: (506) 453-3566
E-mail: fcs@unb.ca

On the equivalence of B-trees and deterministic skip lists

by

Michael G. Lamoureux
Bradford G. Nickerson

University of New Brunswick
Faculty of Computer Science
P.O. Box 4400
Fredericton, N.B., Canada
E3B 5A3

phone: (506) 453-4566 fax: (506) 453-3566
e-mail: y320@unb.ca (Michael G. Lamoureux)
e-mail: bgn@unb.ca (Bradford G. Nickerson)

Abstract

In this paper we show the functional and structural equivalence of B-trees and deterministic skip lists by defining two new B-tree structures, the Bd-tree and Bd$^+$-tree, which we use to provide an explicit mapping between deterministic skip lists (DSLs) and B-trees. This mapping is invertible and formalizes the correspondence that exists between the structures, allowing us to prove the equivalence of B-tree and deterministic skip list data structures. This result is important for a number of reasons. It validates the use of deterministic skip lists as index structures while allowing one to determine the relationship of deterministic skip lists to other balanced search tree structures. This equivalence provides further insight into a comparison of the 1-3 deterministic skip list to the red-black tree and to the 1-d range tree. We now know that we can interchange between the red-black tree and the 1-3 DSL and that the dynamic 1-d range tree and 1-3 DSL are equivalent structures.

Keywords: B-trees, deterministic skip lists, equivalent data structures, balanced search tree structures, Bd-trees, Bd$^+$-trees

TABLE OF CONTENTS

1. Introduction

As noted in [12], there is a correspondence between B-trees and deterministic skip lists. For any B-tree of order m ($m \geq 3$), we are able to define a deterministic skip list (DSL) with a gap size between $\lceil m/2 \rceil - 1$ and m-1 (to provide us with a 1-2, 1-3, 2-4, 2-5, ... DSL) which achieves equivalent worst case cost functions with respect to searching, insertion, and deletion. It was also noted in [10] that there does exist a correspondence between the 1-2 DSL and 2-3 trees and between the 1-3 DSL and 2-3-4 trees. However, this correspondence has never been formalized.

In this paper we introduce the Bd-tree and a natural extension called the Bd^+-tree, which is both functionally and structurally equivalent to the deterministic skip list. We use these structures to define an invertible mapping between B-trees and deterministic skip lists which we use to show the functional and structural equivalence of the two structures.

This result is important for a number of reasons. It validates the use of deterministic skip lists as index structures and lends support to the use of skip lists as database indexes ([5]). It allows one to determine the explicit relationship of deterministic skip lists to other tree structures and, since B-trees are equivalent to red-black trees [4], it is straightforward to determine the relationship between deterministic skip lists and balanced binary search tree structures. For example, we can show that the 1-3 DSL and the dynamic 1-d range tree (as found in [14]) of Bentley [2] are equivalent data structures.

Definition 1:

We define structural equivalence to mean that any structure or sub-structure of one data structure exists in a similar or *equivalent* form in the other data structure. By *equivalent* form we imply that any operation that can be performed on one structure can be performed on the other structure in an analogous fashion. We note that one implication of this definition is that if datum A logically exists next to datum B in structure G_1 then datum A logically exists next to datum B in structure G_2.

Definition 2:

We define functional equivalence to mean that the worst-case cost functions are of the same Big-Oh order of complexity. The following are the six basic cost functions defined on a data structure G of n keys which must be of equal complexity in both data structures if we are to have functional equivalence (based on [11]):

$P(n)$ - preprocessing time required to build G,
$S(n)$ - storage space required by G,
$Q_M(n)$ - time required to search for a single key in G,
$Q_R(n)$ - time required to perform a range search on G,
$I(n)$ - time required to insert a new key into G, and
$D(n)$ - time required to delete an existing key from G.

Requiring the above functions to have the same worst case time cost in both data structures provides not only a necessary condition for functional equivalence but a sufficient one as well. The purpose of a data structure is to index data for storage, processing, and retrieval. S(n) takes care of any necessary storage costs. Processing is handled by the application program, so we need only concern ourselves with the time it takes to search for (and retrieve) the key(s) of interest (which is accounted for by $Q_M(n)$ and $Q_R(n)$) and the time needed to make any necessary updates to the structure (and these costs are covered by P(n), I(n), and D(n)).

We will briefly review deterministic skip lists and B-trees to set the definitions and establish some important terminology before proving the equivalences.

2. Deterministic Skip Lists

Probabilistic skip lists [13], which were introduced as an alternative to balanced tree structures, support range search and have an expected worst case cost for $Q_M(n)$, I(n), and D(n) of O(lg n). However, the worst case is O(n) for all of these cost functions.

Based on the probabilistic skip list of [13], [10] have introduced the deterministic skip list (DSL) which is similar to the probabilistic version but it uses no notions of probability at all. This structure has a worst case search time, $Q_M(n)$, of $O(m\lg_{\lceil m/2\rceil-1}n)$ (where m is the largest gap size) with an equivalent cost for insertion (I(n)) and deletion (D(n)) (and therefore P(n) is $O(mn \lg_{\lceil m/2\rceil-1}n)$. Note that this gives us a worst case range search cost, $Q_R(n)$, of $O(m\lg_{\lceil m/2\rceil-1}n + t)$. Their analysis shows that the DSL space requirements, S(n), are O(n) and that the cost functions are optimal in the 1-d case.

Although skip lists are often represented in the standard form of [13], as in Figure 1, and implemented using *elements*, we often see the linked list representation (which uses *nodes*) introduced in [10] and found in [11], [7], and [8], as shown in Figure 2. Another implementation is the horizontal array implementation [15], a variation of which is shown in Figure 3. In all the Figures that follow, shaded cells contain pointers (which may be nil links) and the pointers always point down or to the right. This convention holds throughout the rest of the paper.

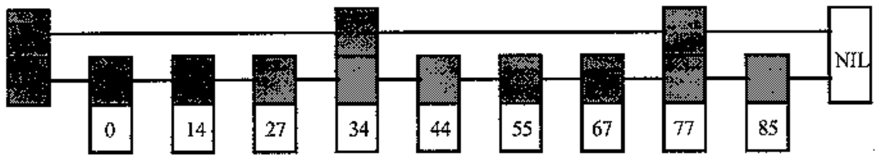

Figure 1. A skip list in standard form.

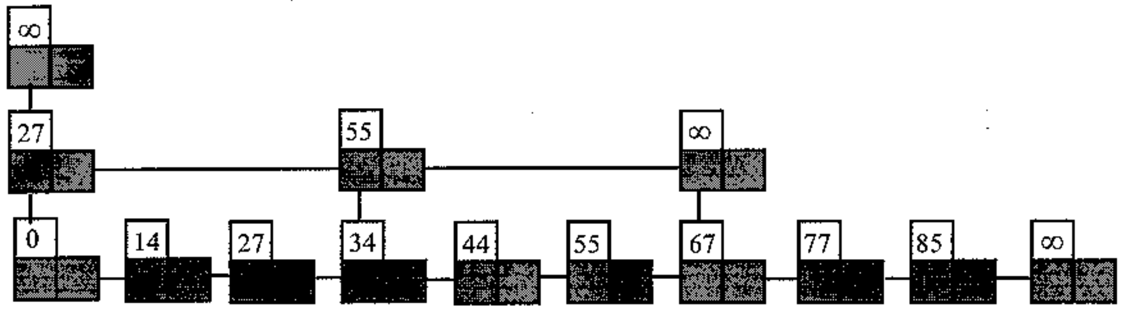

Figure 2. A linked list representation of the skip list of Figure 1.

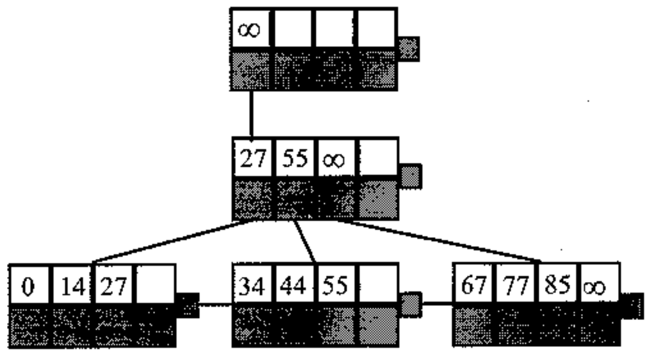

Figure 3. A horizontal array representation of the skip list of Figure 1.

Note that our linked list representation does not use the bottom and tail sentinel nodes found in [10]. They are not necessary in the definition of the structure, although they do simplify coding. Instead of nil links in the down pointers of the nodes on the leaf level we would connect them to the bottom node and instead of nil links in the right pointers of the last node on each level we would connect them to the tail node to simplify coding and implementation.

An *element* (used in the standard implementation represented by Figure 1) is a record that contains a key value and an array of pointers. A *node* (found in the linked list implementation) is a record that contains a key value, a down pointer, and a right pointer. In the standard implementation, the level of an element is the number of (right) pointers it possesses and the level of a node in the linked list implementation is determined with respect to the leaf level which is defined to be level 0. The node structure in the horizontal array implementation is a record structure which contains an array of key values, an array of down pointers, and one right pointer; the level of a node is defined with respect to the leaf level which is defined to be level 0.

The down subtree of a node consists of all nodes that can be reached by traversing the down pointer of the current node such that those nodes are not reachable by traversing the down pointer of the node pointed to by the right pointer (if one exists). In Figure 2, the nodes (0,14, and 27) at level 0 are in the down subtree of node (27) at level 1 and the node (34) at level 0 is not in the down subtree of node (27) at level 1 as it is in the down subtree of node (55) at level 1.

The immediate down subtree of a node is all nodes at level (i-1) reachable by traversing the down pointer of the current node at level i. Thus, the nodes (27, 55, and ∞) at level 1 are in the immediate down subtree of the node (∞) at level 2 and the nodes (67, 77, 85, ∞) are in the immediate down subtree of the node (∞) at level 1. Gap size is defined as one less than the number of nodes in the immediate down subtree.

The direct descendent in the immediate down subtree is that node which is the first node in a node's immediate down subtree. Hence, the gap size can be defined as the number of nodes in the immediate down subtree when we exclude the direct descendent.

Before concluding this section, we would like to note that our variation of the horizontal array implementation of a deterministic skip list is equivalent to a linked list implementation. To see this, we simply replace each node with a linked list of nodes that contain a key value, a down pointer, and a right pointer. The only advantage offered by the horizontal array implementation is direct access to a key value (and associated down pointer) in the immediate down subtree.

3. B-trees

An excellent overview of B-trees can be found in [3]. In summary, a B-Tree of order m has $Q_M(n)$, $I(n)$, and $D(n)$ of $O(m\lg_{\lceil m/2\rceil-1}n)$ (and therefore $P(n)$ of $O(mn\lg_{\lceil m/2\rceil-1}n)$), $Q_R(n)$ of $O(m\lg_{\lceil m/2\rceil-1}n + t)$ (for t the number of keys in range), and $S(n)$ of $O(n)$. We briefly review the basic B-tree structure before we show the equivalence of B-trees and deterministic skip lists. The B-tree can be viewed as a generalization of the height balanced binary search tree (which is the AVL tree of [1]) to a multi-way tree structure in which more than two paths may leave a given node.

A B-tree, T, of order m is defined as an m-way rooted tree which has the following properties (see [3]):

1. Every node, t, has the following fields:
   count: the number of keys (c) currently stored in node t
   keys: an array which holds the key values of node t
   children: an array which holds the pointers to the children of node t
2. Every node t has at least $\lceil m/2 \rceil$ but no more than m children except for the root node which has at least two children if the tree does not consist of the root node alone.
3. All leaves appear at the same level. The depth of a leaf node is the height of the tree.
4. A non-leaf node with c children contains c-1 keys which partition the children in the fashion of a search tree.

The simplest B-tree is a B-tree of order 3. Every internal node than has either 2 or 3 children and we have a 2-3 tree which we will soon see is equivalent to a 1-2 DSL. Thus, a B-tree of order 4 gives us a 2-3-4 tree which we find to be equivalent to the more common 1-3 DSL. In practice, much larger values of m are typically used. Figure 4 illustrates a B-tree of order 4. Note that we follow the convention of using vertical bars to the left and right of key values to represent down pointers (which may be nil links).

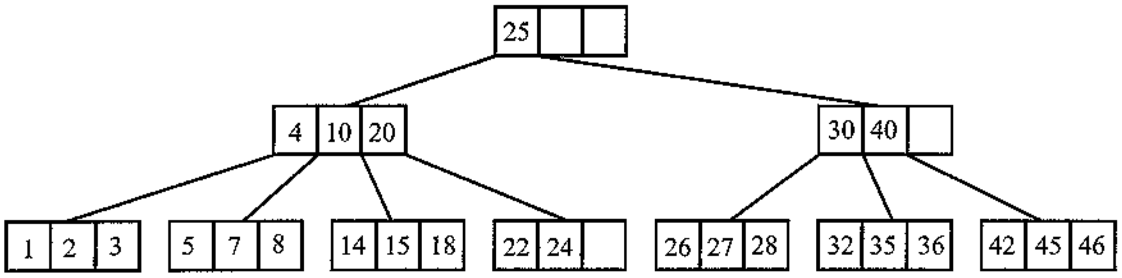

Figure 4. A B-tree of order 4.

4. Bd-trees and Bd$^+$-trees

Before we show the equivalence of B-trees and deterministic skip lists, we must introduce the structures we used to define and accomplish the invertible mapping. The first of these is the Bd-tree. The Bd-tree is essentially a simple B-tree where each node in the structure is connected by a simple (right) pointer to its right sibling, if one exists.

Formally we define the Bd-tree of order m as a modified B-tree of order m where the node structure is modified to contain a "right" pointer which points to the right sibling of the node, if one exists. A Bd-tree corresponding to the B-tree of Figure 4 is shown in Figure 5. The double bar at the right of a node implies the existence of two pointers, one down and one to the right.

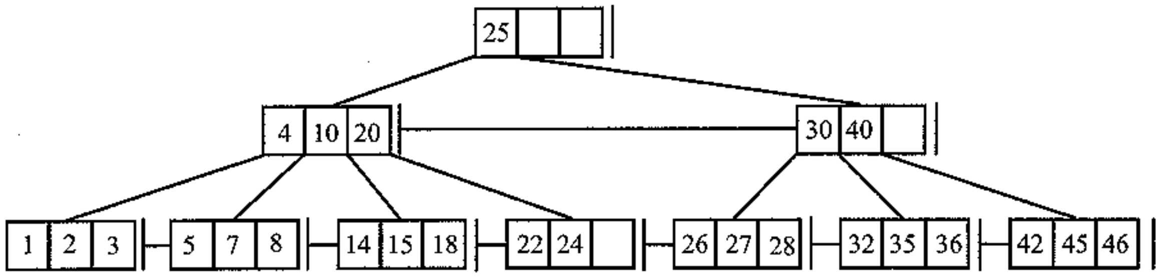

Figure 5. A Bd-tree of order 4 corresponding to the B-tree of order 4 in Figure 4.

We are now able to define the Bd$^+$-tree (of order m) as a modified Bd-tree (of order m) as follows:

1. Each node contains room for m keys and m children.
2. If a node contains c children then it contains c keys.
3. The preceding key from the parent node in the Bd-tree (on the downward path to each node) is added as the rightmost key of the current node of the Bd$^+$-tree.
4. The node consisting of ∞ is attached to the structure and is made the new root.
5. We change property 2 of the B-tree to state: "... the root node has only one child and the only child of the root node (if it exists) has to have at least two children if it has any children at all."

The Bd$^+$-tree is shown in Figure 6. We can see that this structure is both structurally and functionally equivalent to the horizontal array implementation of the deterministic skip list (as seen in Figure 3). The shaded boxes in the horizontal array are collapsed into the vertical bars of the Bd$^+$-tree.

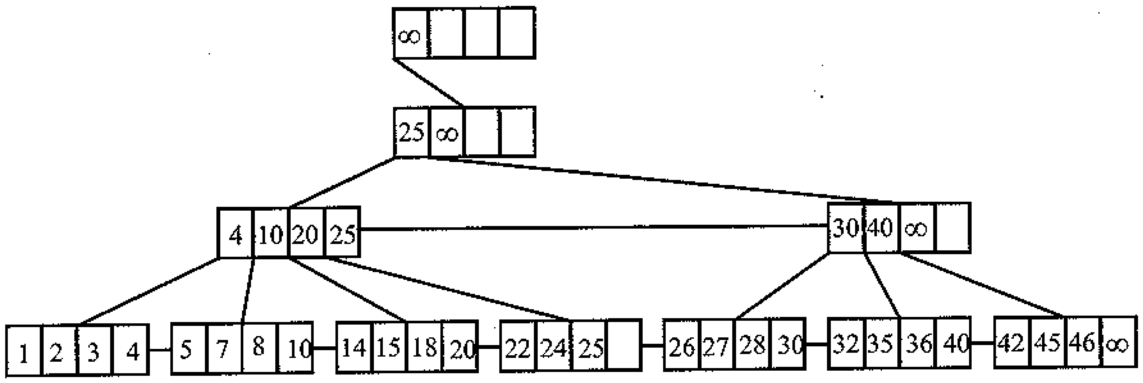

Figure 6. A Bd$^+$-tree of order 4.

One should note the close similarity between the Bd$^+$-tree and the B-link tree of [6] which is a B$^+$-tree (see [3]) where each node is connected to it's right sibling, if one exists. The only difference between the Bd$^+$-tree and the B-link tree is the existence of the "extra" root node which contains only the key value ∞ and the extra key value in each non-leaf node. If we remove the "extra" root node and the extra key value in each non-leaf node then we have the B-link tree of [6]. Figure 7 shows the B-link tree corresponding to the Bd$^+$-tree of Figure 6.

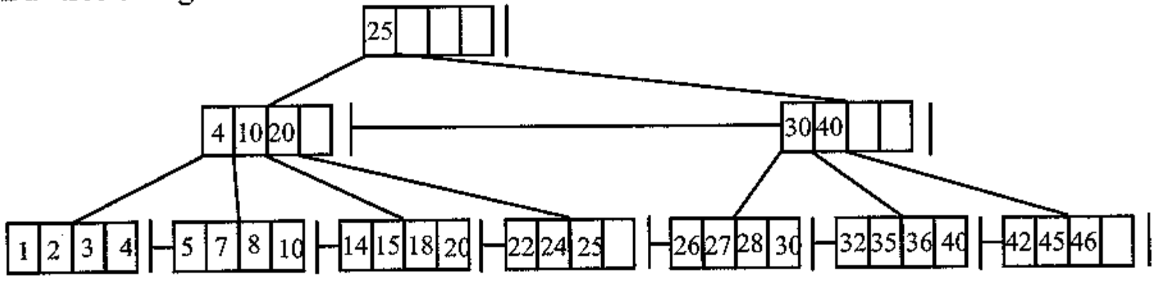

Figure 7. The B-link tree of order 4 corresponding to the Bd$^+$-tree of Figure 6.

5. B-trees and Deterministic Skip Lists

In this section we show that B-trees and deterministic skip lists are equivalent structures.

Theorem 1: A B-tree of order m is structurally equivalent to the linked list implementation of a deterministic skip list of order m.

Proof:

We accomplish the proof by showing that B-trees are equivalent to Bd-trees, that Bd-trees are equivalent to Bd^+-trees, and that Bd^+-trees are equivalent to a horizontal array implementation of a deterministic skip list (which is equivalent to the linked list implementation).

It is straightforward to see that B-trees and Bd-trees are equivalent. The only difference between a B-tree and a Bd-tree is the existence of a right pointer which is used to connect a node to it's right sibling, if one exists. As this right pointer is unnecessary (we can use a recursive traversal procedure to go from a node to its right sibling), the two structures are equivalent.

It is just as straightforward to see that the Bd-tree and Bd^+-tree are equivalent structures. The only difference is the existence of the extra key value in each node which is pulled down from the parent node (and the *extra* root node). As this value can be *remembered* in a recursive implementation of a given traversal procedure (in which the parent is visited before the child node), we see that the two structures are equivalent.

The Bd^+-tree is equivalent to a horizontal array implementation of a skip list as the two are structurally identical. Since the horizontal array implementation of a skip list is equivalent to the linked list implementation (see [15]) of a skip list, we have shown that the B-tree and the linked list implementation of the deterministic skip list are structurally equivalent. ■

Theorem 2: A B-tree of order m is functionally equivalent to the linked list implementation of a deterministic skip list of order m.

We accomplish the proof by showing that B-trees are equivalent to Bd-trees, that Bd-trees are equivalent to Bd^+-trees, and that Bd^+-trees are equivalent to the horizontal array implementation of a deterministic skip list (which is equivalent to the linked list implementation). This is done to be consistent with the proof for Theorem 1.

It is straightforward to see that the B-trees and Bd-trees are equivalent. The only difference is the existence of an additional right pointer at each node of the Bd-tree with a right sibling. This is only a constant increase in storage so S(n) stays the same. Ignoring this pointer allows us to use algorithms designed for the standard B-Tree so I(n) and D(n) (and therefore P(n)) stay the same as do $Q_M(n)$ and $Q_R(n)$. As all cost functions are equivalent, the two structures are functionally equivalent.

It is just as straightforward to see that the Bd-tree and Bd^+-tree are equivalent. In the Bd^+-tree we can ignore the extra key value in each node and skip over the root node and use Bd-tree algorithms. This implies that I(n), D(n) (and therefore P(n)), $Q_M(n)$ and $Q_R(n)$ stay the same. The extra key value and extra root node only increase storage by a constant amount so S(n) stays the same as well. As all cost functions are equivalent, the two structures are functionally equivalent.

Since the Bd^+-tree is structurally equivalent to a horizontal array implementation we can use either set of algorithms on either structure. This implies that if the standard algorithms for working with B-trees (which we use on Bd^+-trees), are of the same order of complexity as the standard algorithms for working with deterministic skip lists then our proof is complete. Since this is true (from comparing the cost functions given in sections 2 and 3), we have shown that the B-tree and the linked list implementation of the deterministic skip list are functionally equivalent. ■

This result allows one to determine the exact relationship between deterministic skip lists and balanced binary search tree structures by using previous results that relate the B-tree to well known balanced binary search tree structures. We can replace the 1-3 DSL with a red-black tree and we can show that the 1-3 DSL is equivalent to the dynamic range tree of [2] (as dynamized by [16] and [9]).

We would like to note that these results are extendible to higher dimensions. For example, we can show that the k-d range tree of [2] as dynamized by [16] and [9] is equivalent to the k-d Range DSL of [7]. This result is illuminated in [7] where the two structures are shown to have the same cost functions.

6. Conclusions

In this paper we have formalized the relationship between deterministic skip lists and B-trees. We have shown that the linked list implementation of a deterministic skip list is structurally and functionally equivalent to the well known B-tree data structure. This result is important as it validates the use of deterministic skip list structures as index structures (on databases) and allows one to determine the exact relationship between deterministic skip lists and balanced binary search tree structures.

Along the way we have defined two more alternative B-tree structures which could prove faster in implementation as the existence of right pointers eliminates the need for backtracking and heavy recursion (by allowing for more iterative procedures). Also, the existence of the extra key value in each node (pulled down from the parent) in the Bd^+-tree eliminates the need to remember the previous key.

This work is extendible to higher dimensions and thus one could not only choose between using a multidimensional B-tree or k-d deterministic skip list as an index structure for a multidimensional database, but define a corresponding k-d deterministic skip list (multidimensional B-tree) structure given a multidimensional B-tree (k-d deterministic skip list) structure to start with. This equivalence gives one a high degree of flexibility when choosing a data structure according to well-defined functional requirements.

7. Acknowledgement

The authors would like to thank the Faculty of Computer Science at the University of New Brunswick for the financial and infrastructure support provided during the course of this research.

8. References

[1] Adel'son-Vel'skii, G.M., and Landis, E.M., "An Algorithm for the Organization of Information", *Soviet Mathematics Doklady*, Vol. 3 (1962), pp. 1259 - 1262.

[2] Bentley, J.L., "Multidimensional Binary Search Trees Used for Associative Searching", *Communications of the ACM*, Vol. 18, No. 9 (1975), pp 509 - 517.

[3] Comer, Douglas, "The Ubiquitous B-Tree", *Computing Surveys*, Vol. 11, No. 2, June 1979, pp. 121 - 137.

[4] Guibas, Leo J. and Sedgewick, Robert, "A Dichromatic Framework for Balanced Trees", Proceedings of the 19th Annual Symposium on Foundations of Computer Science, Ann Arbor, Michigan, Oct 16-18, 1978, pp. 8 - 21.

[5] Hanson, Eric N. and Johnson, Theodore, "Selection Predicate Indexing for Active Databases Using Interval Skip Lists", Computer and Information Sciences Department of the University of Florida Technical Report, TR94-017 (April 15,1994).
(anonymous ftp site "ftp.cis.ufl.edu" in file "/cis/tech-reports/tr94/tr94-017.ps.gz")

[6] Johnson, Theodore, "A Highly Concurrent Priority Queue Based on the B-link Tree", Electronic Tech Report #007-91, University of Florida, Dept. of CS (Aug 18, 1991).
(anonymous ftp site "ftp.cis.ufl.edu" in file "/cis/tech-reports/tr91/tr91-007.ps.Z")

[7] Lamoureux, Michael G. and Nickerson, Bradford G., "Deterministic Skip Lists for k-dimensional Range Search", University of New Brunswick Technical Report TR95-098, October 1995. (94 pages)
(anonymous ftp site "ftp.unb.ca" in file "pub/factulty.cs/trs/tr95-098.ps")

[8] Lamoureux, Michael G. and Nickerson, Bradford G., "Deterministic Skip List Data Structures: Efficient Alternatives to Balanced Search Trees", to appear in the 19th Annual Atlantic Mathematics and Computing Science Days (APICS '95) conference proceedings, Sydney, Cape Breton (Nova Scotia, Canada), Oct 20-21, 1995.

[9] Lueker, Geroge S., "A Data Structure for Orthogonal Range Queries", Proceedings of the 19th Annual Symposium on Foundations of Computer Science, IEEE 78 CH 1387-9 C, pp 28-34.

[10] Munro, J. Ian; Papadakis, Thomas; & Sedgewick, R, "Deterministic Skip Lists", Proceedings of the Third Annual Symposium on Discrete Algorithms, Orlando, Florida, January 27-29, 1992.

[11] Nickerson, Bradford G., "Skip List Data Structure for Multidimensional Data", Comp. Sci. Technical Report Series, CS TR-3262 UMIACS CS-TR-94-52, April 1994. (39 pages) (anonymous ftp "ftp.cs.umd.edu" in file "/pub/papers/TRs/3262.ps")

[12] Papadakis, T., "Skip Lists and Probabilistic Analysis of Algorithms", Ph.D. Thesis, University of Waterloo, 1993.
(anon. ftp site "cs-archive.uwaterloo.ca" in file "cs-archive/cs-93-28")

[13] Pugh, William, "Skip Lists: A Probabilistic Alternative to Balanced Trees", *Communications of the ACM*, June 1990, Volume 33, Number 6 pp. 668 - 676.

[14] Samet, H., *The Design and Analysis of Spatial Data Structures*, Addison-Wesley Publishing, Reading, MA., 1990.

[15] Weiss, Mark Allen, *Data Structures and Algorithm Analysis*, Benjamin/Cummings Publishing Company, Inc., Redwood City, California, 1994.

[16] Willard, Dan E., "New Data Structures for Orthogonal Range Queries", *SIAM J. Comput.*, Vol. 14, No. 1, (February) 1985, pp. 232-253.