International Journal Of Engineering Research & Management Technology ISSN: 2348-4039

September- 2014 Volume 1, Issue-5

Email: editor@ijermt.org www.ijermt.org

Finger Search Trees, Skip Lists, Treaps

Riya Sunidhi Rohit Tanwar
Department of Information Technology
Dronacharya College of Engineering
Gurgaon, Haryana

ABSTRACT:
Finger searching is a good way to facilitate efficient search. A finger search beginning at a particular element of an array is done by an exponential search by inspecting the elements at a distance of 2^i-1 from the finger. Alternative to finger search trees are treaps and skip lists. A treap satisfies elements which are sorted to inorder traversal of tree. A skip list is considered to be that kind of a data structure that consists of expected (log n) levels. Its certain main applications are that it is used in computational geometry, set operations, for list splitting, for adaptive merging and sorting.

Keywords: Finger search, skip lists, treaps

I. INTRODUCTION

A major problem in computer science is to maintain a sorted sequence of elements to facilitate efficient searches. One of the best solutions to the problem is to maintain a sorted sequence in the form of balanced search tree to enable insertions, deletions and searches in logarithmic time.
A finger search tree are basically search trees that support fingers, hence, pointers to elements in the search trees and supports efficient updates and searches in the vicinity of the fingers. Skip lists and treaps are basically applications of finger search trees.

II. DEFINATION

A finger search on a data structure is an extension of any search operation that supports structure and a reference (finger) to an element in the data structure is given along with the query. Finger search trees are a function of distance between element and the finger.

III. DYNAMIC FINGER SEARCH TREES

A dynamic finger search tree in addition to finger search trees supports the addition and deletion of elements on a position given by a finger. These finger search trees assume either a fixed constant number of fing
Guibas et al had introduced finger search trees as a variant of B-trees which supports finger search trees in 0(log d) time. Moving a finger for d positions requires 0(log d) time. This theory was further modified by Huddleston and Mehlhorn. Tarjan and van Wyk presented a solution based on red-black trees.

Constructions that support 0(log d) time searches and 0(log* n) time insertions and deletions were developed by Harel and Fischer. Finger search trees with worst-case constant time insertions and 0(log*n) was presented by Brodal et al.
Splay trees were introduced as a class of self-adjusting binary search trees by Slater and Trajan. It supports searches, insertions, deletions in amortized 0(log n)time. These splay trees, later proved by Cole, were quite efficient finger search trees.

An 0(n) initialization cost ,the amortized cost, the amortized cost of an access at distance d from preceding access in a splay tree is 0(log d)where accesses include insertion, deletion and searches. The above statement applies in presence of only one finger that points the last accessed element.
The above mentioned constructions could only be implemented on a pointer machine where the only operation is that of comparison of two elements. In order to overcome this, Random Access Machine model of computation, was developed by Dietz and Raman. They developed a finger search tree with constant update time and 0(log d) search time. In the same model of computation, a logarithmic bound in search procedure query time $(logd/log\ logd)^{1/2}$ was achieved.

I. LEVEL-LINKED (2,4) TREES

A (2,4)-tree is a height balanced search tree where all leaves are of same height and depth. All internal nodes have degree of two, three or four. Internal nodes only store search keys for guiding searches and all elements are stored at leaves. As each internal node at least a degree of two, it has a height of 0(log n) and supports searches in 0(log n) time.

A (2,4)-tree that has n leaves can be split to two trees n_1 and n_2 in amortized $0(\log \min(n_1,n_2))$ time. In order to support finger search (2,4)-trees are augmented with level links equal depth are linked together in a double linked list. For performing a finger search from x to y ,we first check if y is to the left or right of x. Assume without loss of generality that y is to the right of x. Traversing of path is done from x towards the root while examining the nodes on the path and their neighbors until it has been established that y is contained within sub-tree rooted at v or v's right neighbor .Upwards search is terminated and at two most downwards searches for y is started respectively at v or v's right neighbor. The 0(log d) search time follows from the observation that if we advance the upwards search to the node to the parent of node v then y is to right of leftmost sub tree of v's right neighbor.

II. SKIP LISTS

A randomized dictionary data structure which consists of expected 0(log n)levels is called a skip list. The lowest level present in a linked list contains elements in sorted order and each succeeding level is a random sample of elements of previous level, where each element is included in the next level with a fixed probability.
The most considered properties of a skip list are that they support insertions and deletions at a given position, require expected linear space consist of expected 0(log n)levels and support searches in expected 0(log n)time.

To facilitate backward finger searches, a finger to a node v is stored as an expected 0(log n) space finger data structure that for each level i stores a pointer to the node to the left of v where the level i pointer either points to v or a node to the right of v.

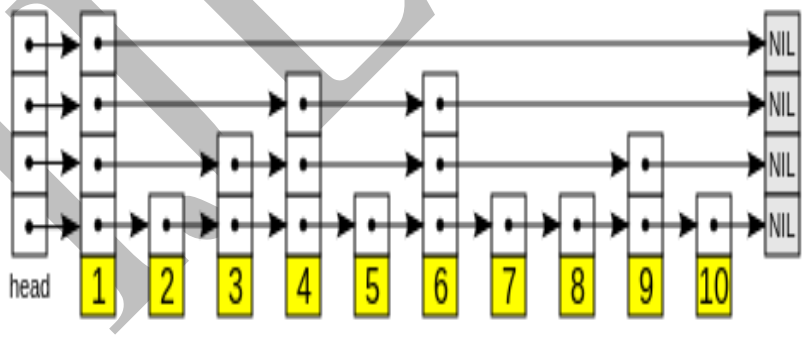

Fig: Finger search on skip list

III. TREAPS

A treap is an important application of finger search tree. A rooted binary tree in which each node stores an element where each element has an associated random priority is called a treap. It satisfies that the elements are sorted with respect to an in order traversal of tree, the priority elements satisfy heap order ,i.e. the priority stored at a node is always smaller than or equal to the priority stored at the parent node. The most prominent properties of treaps are

- Expected 0(log n) height, implying that they provide searches in expected log n time.
- insertion of elements
- deletion of elements

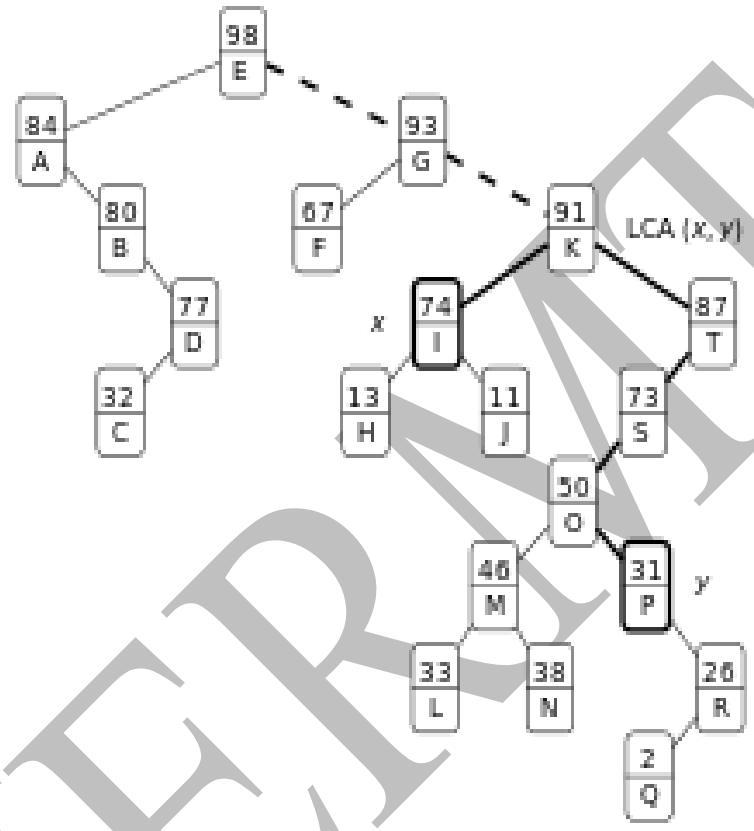

Fig: Performing finger search on treap

The essential property of treaps which enable 0(log d)finger searches is that for elements x and y whose ranks differ by d in the set stored, the expected length of path between x and y in treap is 0(log d). In order to perform a finger search for y starting at a finger at x, we ideally start at x and traverse the path till we reach the least ancestor path of x and y and start a downward search for y.

ACKNOWLEDGEMENT

We thank our teacher for constantly helping us to write this research paper. Our teacher is greatly to be thanked who suggested us this topic. We thank our parents for supporting us throughout our writing.

REFERENCES

1. Gerth Brodal Univesity of Aarhus G. S. Brodal. Finger search trees with constant insertion time. In Proc. 9th Annual ACM-SIAM Symposium on Discrete Algorithms, pages 540–549, 1998.
2. G. S. Brodal and R. Jacob. Dynamic planar convex hull. In Proc. 43rd Annual Symposium on Foundations of Computer Science, pages 617–626, 2002.
3. G. S. Brodal, G. Lagogiannis, C. Makris, A. Tsakalidis, and K. Tsichlas. Optimal finger search trees in the pointer machine. Journal of Computer and System Sciences,
4. Special issue on STOC 2002, 67(2):381–418, 2003. G. S. Brodal, R. B. Lyngsø, C. N. S. Pedersen, and J. Stoye. Finding maximal pairs with bounded gap. Journal of Discrete Algorithms, Special Issue of Matching Pat-terns, 1(1):77–104, 2000.
5. G. S. Brodal and C. N. S. Pedersen. Finding maximal quasiperiodicities in strings. In 11-10 Proc. 11th Annual Symposium on Combinatorial Pattern Matching, volume 1848 of Lecture Notes in Computer Science, pages 397–411. Springer Verlag, 2000.
6. M. R. Brown and R. E. Tarjan. A fast merging algorithm. Journal of the ACM, 26(2):211–226, 1979.
7. M. R. Brown and R. E. Tarjan. Design and analysis of a data structure for representing sorted lists. SIAM Journal of Computing, 9:594–614, 1980.
8. S. Carlsson, C. Levcopoulos, and O. Petersson. Sublinear merging and natural merge-sort. Algorithmica, 9(6):629–648, 1993.
9. R. Cole. On the dynamic finger conjecture for splay trees. part II: The proof. SIAM Journal of Computing, 30(1):44–85, 2000.
10. R. Cole, B. Mishra, J. Schmidt, and A. Siegel. On the dynamic finger conjecture for splay trees. part I: Splay sorting log n-block sequences. SIAM Journal of Computing, 30(1):1–43, 2000.
11. P. F. Dietz and R. Raman. A constant update time finger search tree. Information Processing Letters, 52:147–154, 1994.
12. V. Estivill-Castro and D. Wood. A survey of adaptive sorting algorithms. ACM Computing Surveys, 24:441–476, 1992.
13. R. Fleischer. A simple balanced search tree with O(1) worst-case update time. Inter- national Journal of Foundations of Computer Science, 7:137–149, 1996.
14. L. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. Tarjan. Linear time algorithms for visibility and shortest path problems inside simple polygons. Algorithmica, 2:209–233, 1987.
15. L. J. Guibas, E. M. McCreight, M. F. Plass, and J. R. Roberts. A new representation 717896933-indianbankchennaia