Journal of Automata, Languages and Combinatorics 8 (2003) 4, 607–632
© Otto-von-Guericke-Universität Magdeburg

NEW DYNAMIC BALANCED SEARCH TREES WITH WORST-CASE CONSTANT UPDATE TIME [1]

George Lagogiannis, Christos Makris, Yannis Panagis,
Spyros Sioutas and Kostas Tsichlas
*Computer Engineering and Informatics Department, University of Patras
Patras, Greece*
e-mail: {lagogian, makri, panagis, sioutas, tsihlas}@ceid.upatras.gr

ABSTRACT

We present new search trees with worst-case $O(1)$ update time and $O(\log n)$ search time, storing n elements in linear space, in the Pointer Machine (PM) model of computation. In addition, these trees can easily support finger searches in time $O(\log d)$ and update operations in worst-case $O(\log^* n)$ time. The parameter d represents the number of elements (distance) between the search element and an element pointed to by a pointer termed *finger*. Our data structure is based on a previous result by Fleischer that exhibits the same asymptotic time and space complexities for simple search trees. We improve on this result by handling deletions in an explicit way without using the standard trick of global rebuilding. This is the first search tree that combines worst-case update times with a local rebalancing scheme without using global rebuilding to tackle deletions. In addition, insight is acquired from the construction of these trees as to why deletions are considered more difficult than insertions in the (a,b)-trees setting. Finally, we hope that these techniques may lead to a simpler version of the constant update finger search tree presented recently by Brodal et al. [5].

Keywords: Balanced trees, update operations, data structures, complexity

1. Introduction

The *balanced search tree* is one of the most common data structures used in algorithms. They constitute an elegant solution to the elementary *dictionary* problem, in which one needs to maintain a dynamic set of elements under the operations of insertion and deletion so that the search for a particular element is carried out efficiently. In general, known balanced search trees like AVL-trees [1], red-black trees [20] and (a,b)-trees [13] support search and update operations in $O(\log n)$ time when storing n elements. In the Pointer Machine (PM) [2, 19] model of computation the time for the search procedure cannot be reduced since the lower bound of $\Omega(n \log n)$ for sorting n elements would be violated. Since update operations are preceded by a search operation, their time complexity is also $\Omega(\log n)$. However, in the bibliography there

[1]Full version of a lecture presented at the *Thirteenth Australasian Workshop on Combinatorial Algorithms* (Kingfisher Bay Resort, Fraser Island, Queensland, Australia, July 7 – 10, 2002).