RAIRO. INFORMATIQUE THÉORIQUE

ATHANASIOS K. TSAKALIDIS

# Rebalancing operations for deletions in AVL-trees

*RAIRO. Informatique théorique*, tome 19, n° 4 (1985), p. 323-329

<http://www.numdam.org/item?id=ITA_1985__19_4_323_0>

R.A.I.R.O.Informatique théorique/Théoretical Informatics
(vol. 19, n° 4, 1985, p. 323 à 329)



# REBALANCING OPERATIONS FOR DELETIONS IN AVL-TREES (*)

by Athanasios K. TSAKALIDIS (1)

Communicated by W. BRAUER

Abstract. — *Considering only deletions performed in an AVL-tree with n leaves, we show that the total rebalancing time for n arbitrary deletions is linear. More precisely, the number of balance and structural changes is bounded by* 1.618 *n.*

Résumé. — *En regardant seulement des suppressions exécutées dans un arbre du genre AVL avec n feuilles on va montrer que le temps nécessaire pour rebalancer l'arbre après n suppressions choisies par hasard est linéaire. Plus précisément le nombre de changements de balance et structurels est limité par* 1.618 *n.*

## 1. INTRODUCTION

Balanced trees (e. g. 2-3 trees, B-trees, AVL-trees) are very popular data structures for the set manipulation problem. The AVL-tree is the *oldest basic* data structure introduced 1962 by Adel'son-Vel'skii and Landis [1]. Many attempts were undertaken to analyse these trees (Foster [3], Knuth [5], p. 455). Brown [2] and Mehlhorn [6] studied the expected number of balanced nodes in random AVL-trees. Mehlhorn and Tsakalidis [8] give a rigorous analysis for insertions into an initially empty AVL-tree. Besides other results it is shown in [8] that the number of rebalancing operations (= balance changes) for insertions is linear.

In this paper we consider only deletions in an arbitrary AVL-tree with $n$ leaves and we show that the number of rebalancing operations (= balance and structural changes) is linear for deletions too. Note that a deletion can cause $O(\log n)$ rebalancing operations. In the case of an insertion the number of structural changes is at most $O(1)$; this is not true for a deletion and thus these are considered as rebalancing operations.

(*) Received July 1984, revised February 1985.
(1) FB 10, Informatik, Universität des Saarlandes, 6600 Saarbrücken, R.F.A.

R.A.I.R.O. Informatique théorique/Theoretical Informatics
0399-0540/85/04 323 07/$ 2.70/

Our main insight is to concentrate on *amortized* behavior rather than expected behavior. This measure is well established in the literature (Mehlhorn [7]) and led us to stronger results (our results hold for arbitrary not just random sequence of deletions) and suggested to use combinatorial (not probabilistic) methods of analysis. More precisely we show, that the total number of rebalancing operations in processing a sequence of $n$ arbitrary deletions from an AVL-tree with $n$ leaves is bounded by 1,618 $n$. Experimental data (Karlton *et al.* [4]) suggests that the expected number of rebalancing operations for $n$ random deletions is 1,126 $n$ and hence only *slightly less* than the amortized number.

## 2. DEFINITIONS AND ELEMENTARY OPERATIONS

AVL-trees are binary trees in which nodes either have two sons or no sons. The latter nodes are called leaves. A binary search tree is AVL if the heights of the subtrees at each node differ by at most one, where the height Height $(v)$ of node $v$ is equal to the length of the longest path from $v$ to a leaf.

Let $L(v)[R(v)]$ be the left [right] subtree of the tree with root $v$. For every node $v$ we define its height balance $hb(v)$ by:

$$hb(v) = \text{Height}(R(v)) - \text{Height}(\text{L}(\text{v})).$$

Hence the height balance can be $+1$, 0, or $-1$. We call a node balanced (unbalanced) if its height balance is $O$ $(\pm 1)$.

For every deletion we define the *critical node* (*CN*) as the node on the search path where the balance changes after the deletion either cause no more height decreases of the subtrees or produce a height balance $+2$ or $-2$ (causing structural changes). We give the last definition more formally:

Let $v_0, v_1, \ldots, v_k$ be the path from the root $v_0$ of an AVL-tree to $v_k$, the leaf searched for. Let $v_s$ be a node on the search path with Height $(v_s)=2$. Then the *critical node* is $v_s$ if either $hb(v_s)=0$ (case 1), otherwise (case 2) the node $v_{j-1}$, where $j$ is minimal such that:

$$hb(v_t) = \begin{cases} +1, \text{ if right-son } (v_t) \text{ lies on the search path,} \\ -1, \text{ if left-son } (v_t) \text{ lies on the search path,} \end{cases}$$

for all $t$ with $j \leqq t \leqq s$, if such a sequence exists or the father of $v_s$ (case 3) otherwise.

(Figure 1 illustrates the three cases up to left-right symmetry.)

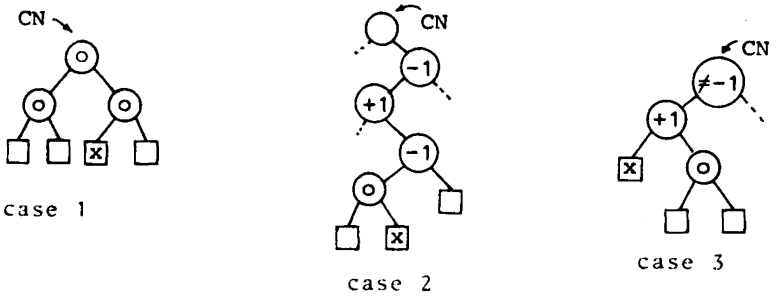



**Figure 1**

In the above figure $x$ is the leaf which must be deleted. In the following figures a node contains its height balance and a subtree is represented by its height $h$.

Next we give the elementary operations executed on the critical node ($CN$) after a deletion.

Let $v$ be the $CN$ with $hb(v) = +1$ as in figure 2. We explore the case where a deletion causes a height decrease of the left subtree of $v$ :

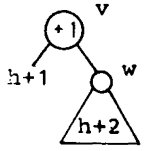



**Figure 2**

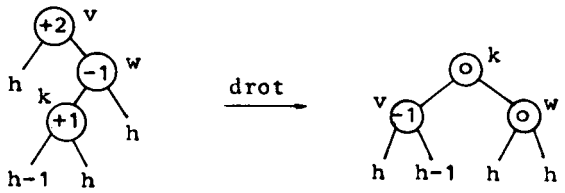
**a.1. Terminating rotation:** $hb(w) = 0$


REMARK a.1: This operation increases the number of the unbalanced nodes by one. The height of the rebalanced subtree remains unchanged.

**a.2. Propagating single rotation:** $hb(w) = +1$:

**a.3. Propagating double rotation:** $hb(w) = -1$

$a.3.1$: $hb(k) = +1$:




$a.3.2$: $hb(k) = 0$, then we get $hb(k) = 0$, $hb(v) = 0$, $hb(w) = 0$.

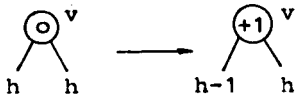
$a.3.3$: $hb(k) = -1$, *then we get* $hb(k) = 0$, $hb(v) = 0$, $hb(w) = +1$.

REMARK $a.2$-3: The operations $a.2$ and $a.3$ each decrease the number of the unbalanced nodes by 2.

The rebalanced subtree decreases its height by 1.

**b. Absorption**

If $hb(v) = 0$ and the subtree of $v$ has decreased its height by one, then we take $hb(v) \neq 0$ without changing any other height:




REMARK *b*: This operation increases the number of the unbalanced nodes by one.

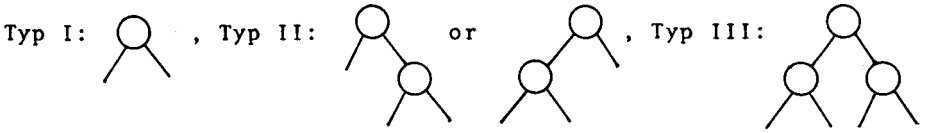
## 3. THE COMPLEXITY OF A SEQUENCE OF DELETIONS

We want to estimate the complexity of $n$ arbitrary deletions in an arbitrary AVL-tree $T_n$ with $n$ leaves. First we give the deletion algorithm:

*proc* delete ($x$);
1) search for $x$;
2) delete $x$;
3) execute the balance changes $\pm 1 \rightarrow 0$ on the search path up to the *CN*;
4) execute *either* one of the operations $a.1$ or *b or* one or more of the propagating operations $a.2$ or $a.3$;
*end*

Let $T$ be an AVL-tree. The *fringe* of $T$ is obtained by deleting all nodes which have more than four leaves below them. The fringe of an AVL-tree is

partitioned into subtrees of the following three types:




A deletion can increase or decrease the number of the unbalanced nodes. We observe the following:

REMARK *c*: A deletion on type III subtree provides a type II subtree and increases the number of unbalanced nodes by one, and a deletion on type II decreases this number by one.

During the execution of $n$ arbitrary deletions on $T_n$ *the following values will become important:*

$X_1$ : the total number of balance changes $\pm 1 \to 0$ on step 3);

$X_2$ : the total number of terminating rotations;

$X_3$ : the total number of propagating single rotations;

$X_4$ : the total number of propagating double rotations;

$X_5$ : the total number of absorptions.

In the worst case for a single deletion the balance changes $\pm 1 \to 0$ and the propagating rotations together could cause cost proportional to the height of the tree, and we should expect that the total cost of the rebalancing operations is $O(n \log n)$. We will show, however, that this cost is only $O(n)$.

THEOREM: *Let $T_n$ be an arbitrary AVL-tree with n leaves, and let REB be the total number of balance changes $\pm 1 \to 0$ on the search path up to the critical node* plus *the total number of the propagating single or double rotations during the execution of n arbitrary deletions on $T_n$, then the following holds:*

$$REB \leqq 1{,}618\ n.$$

*Proof:* For this process we define:

$D$ : = the number of the destroyed unbalanced nodes;

$E$ : = the number of the existing unbalanced nodes in $T_n$;

$P$ : = the number of the newly produced unbalanced nodes.

Since we finally have no nodes any more the following holds:

$$D = E + P. \tag{1}$$

Let $d_1$, $d_2$, $d_3$ be the number of the $n$ deletions which are executed in a subtree of type I, II, III respectively.

Next we estimate the contribution of the variables $d_i$ for $1 \leqq i \leqq 3$ and $X_j$ for $1 \leqq j \leqq 5$ to the number of the unbalanced nodes during this process.

Since a deletion in a subtree of type I destroys a balanced node and always leads either to an operation $a.1$, $a.2$ or $a.3$ the respective contribution of $d_1$ *to the number of the unbalanced nodes will be charged to the variables* $X_2$, $X_3$ or $X_4$. Analogously a deletion in a subtree of type III destroys a balanced node and leads to an operation $b$ (Absorption) and thus the contribution of $d_3$ to the mentioned number will be charged to the variable $X_5$.

*Hence we have only to consider the contribution of* $d_2$ and $X_j$ for $1 \leqq j \leqq 5$ to the number of the unbalanced nodes occured during the $n$ deletions on $T_n$.

According to the remarks $a.1$, $a.2$-3, $b$ and $c$ we get for (1):

$$d_2 + 2.(X_3 + X_4) + X_1 = UN(T_n) + X_2 + X_5, \tag{2}$$

where $UN(T_n)$ is the number of the unbalanced nodes in $T_n$.

According to Knuth ([5], exercise 6.2.3.3) we have:

$$UN(T_n) \leqq 0{,}618\ n. \tag{3}$$

Since the operations $a.1$ and $b$ are terminating we get:

$$X_2 + X_5 \leqq n. \tag{4}$$

Setting (3) and (4) in (2) we get:

$$d_2 + 2.(X_3 + X_4) + X_1 \leqq 1{,}618\ n.$$

Since $d_2 + X_3 + X_4 \geqq 0$ we have:

$$REP = X_3 + X_4 + X_1 \leqq 1{,}618\ n. \quad \square$$

Experiments in [4] show that the average size of $X_1$ is 0,912 and of $X_3 + X_4$ is 0,214 for a random deletion.

## 4. CONCLUSIONS

We have shown that the total number of rebalancing operations after $n$ arbitrary deletions in an AVL-tree with $n$ leaves is at most 1,618 $n$.

For arbitrary mixed insertions and deletions this number cannot be linear since there are interchanged deletions and insertions which always cause rebalancing up to the root.

But for random mixed insertions and deletions such a result seems to be likely.

## REFERENCES

1. G. M. ADEL'SON-VEL'SKII and E. M. LANDIS, *An Algorithm for the Organization of Information*, Dokl. Akad. Nauk S.S.S.R., Vol. 146, 1962, pp. 263-266 (in Russian); English Translation in Soviet. Math., Vol. 3, pp. 1259-1262.
2. M. R. BROWN, *A Partial Analysis of Random Height-Balanced Trees*, S.I.A.M. J. Comput., Vol. 8, 1979, pp. 33-41.
3. C. C. FOSTER, *Information Storage and Retrieval Using AVL-Trees*, ACM 20th National Conference, 1965, pp. 192-205.
4. P. L. KARLTON, S. H. FULLER, R. E. SCROGGS and E. B. KAEHLER, *Performance of Height-Balanced Trees*, Comm. ACM., Vol. 19, 1976, pp. 23-28.
5. D. E. KNUTH, The Art of Computer Programming, *Sorting and Searching*, Vol. 3, Addison-Wesley, Reading, MA, 1973.
6. K. MEHLHORN, *A Partial Analysis of Height-Balanced Trees Under Random Insertions and Deletions*, S.I.A.M. J. Comput., Vol. 11, 1982, pp. 748-760.
7. K. MEHLHORN, *Data Structures and Algorithms* 1, 2, 3, Springer Verlag, E.A.T.C.S. Monographs in Theoretical Computer Science, 1984.
8. K. MEHLHORN and A. K. TSAKALIDIS, *An Amortized Analysis of Insertions into AVL-Trees*, S.I.A.M. J. Comput. (in press).