Amortized Analysis of (a, b)-Trees

Gerth Stølting Brodal and Rolf Fagerberg

BRICS
Department of Computer Science
University of Aarhus

Draft Manuscript

September 7, 2000

1 (a, b)-trees

In the following we consider the class of height balanced trees denoted (a, b)-trees. For the special $b = 2a - 1$, (a, b)-trees are also known as B-trees. More extensive descriptions of (a, b)-trees can be found in standard text books, e.g. [1, 2].

In the following we let the *depth* of a node in a tree be the number of edges on the shortest path from the node to the root, i.e. the root has depth zero. The *degree* of a node is the number of children of the node. The *height* of a tree is the maximal depth of a leaf in the tree, and the height of a node in a tree is the height of the subtree rooted at the node. We let $\log x$ denote the logarithm with base two of x.

Definition 1 *Let a and b be integers where $a \geq 2$ and $b \geq 2a - 1$. A tree is an (a, b)-tree if and only if*

1. *All leaves have the same depth.*
2. *All internal nodes have degree at most b.*
3. *All internal nodes except the root have degree at least a.*
4. *The root has degree at least two.*

From the above constraints of the structure of an (a, b)-tree we have the following bounds on the height of an (a, b)-tree.

Theorem 1 *An (a, b)-tree with n leaves has height between $\left\lceil \frac{\log n}{\log b} \right\rceil$ and $\left\lfloor \frac{(\log n)-1}{\log a} \right\rfloor + 1$.*

Proof. For a tree with n leaves and height h we have by Definition 1 that $2 \cdot a^{h-1} \leq n \leq b^h$, and the theorem follows. □

An (a, b)-tree can be used to store a sorted set, by storing the elements of the set at the leaves in sorted order from left-to-right, and by letting each internal node of degree k store k pointers to its children and $k - 1$ keys, such that i-th key is larger than all elements stored in the subtree rooted at the i-th child and smaller than or equal to all elements stored in the subtree rooted at the $(i + 1)$-st child. Figure 1 shows a $(2, 4)$-tree having 10 leaves and height

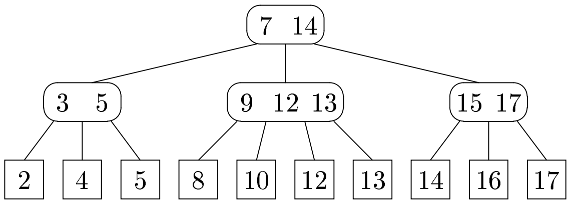

Figure 1: A $(2, 4)$-tree.

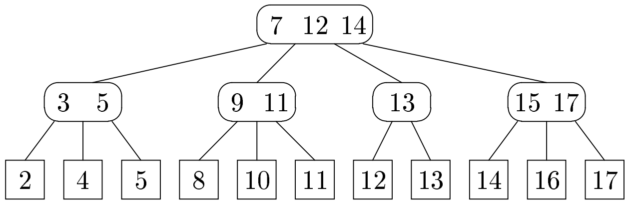

Figure 2: The result of inserting 11 into the $(2, 4)$-tree in Figure 1.

two. Searching for a leaf can be done by a top-down traversal of the path from the root to the appropriate leaf by using the stored keys. By performing a binary search among the keys at each node this takes time $O(\log b \cdot \frac{\log n}{\log a})$, which is $O(\log n)$ for $b = a^{O(1)}$.

2 Inserting and deleting leaves

Important operations on (a, b)-trees are the insertion and deletion of leaves. After having inserted a leaf into or deleted a leaf from an (a, b)-tree, i.e., created or removed a child of a node with height one, the tree might not satisfy the constraints on the shape of an (a, b)-tree.

If an insertion causes the node with height one to get $b + 1$ children, we perform a *node splitting* (see Figure 3). First a new node is created which is made the sibling of the node with $b+1$ children (creating a new root if necessary). The $b+1$ children are then distributed evenly among the two siblings such that they get respectively $\lfloor (b + 1)/2 \rfloor$ and $\lceil (b + 1)/2 \rceil$ children. Note that the degree of the parent increases by one. If the parent now has degree $b + 1$, a recursive node splitting is performed at the parent.

If a deletion causes the node with height one to get $a - 1$ children and the node is not the root, we either perform a *node sharing* or *node fusion* (see Figure 3). If the node is not the root it must have a parent with at least two children, i.e., there exists an immediate sibling to the left or right with between a and b children. If the sibling has at least $a + 1$ children, the closest child of the sibling is moved to the node with $a - 1$ children such that the node gets a children (sharing). If the sibling has a children we fusion the two siblings into one node with $2a - 1$ children. This decreases the number of children at the parent by one (fusion). If the parent now has $a - 1$ children and is not the root we recursively perform a node fusion or node sharing at the parent. If the root after the deletion has only one child, the root is deleted.

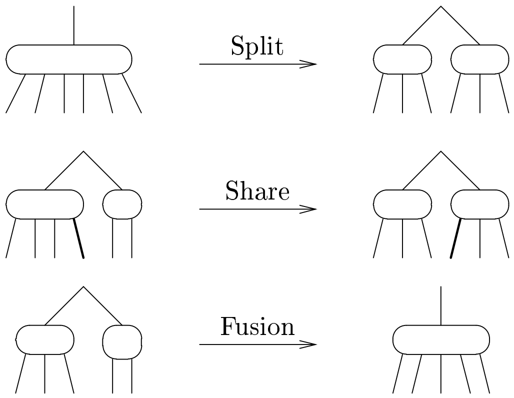

Figure 3: Node splitting, node fusion, and node sharing in $(2,5)$-trees.

3 Analysis

We first analyze the case where no deletions are performed on an (a,b)-tree. This implies that all rebalancing is done by node splittings.

Theorem 2 *In a sequence of n insertions on an initial empty (a,b)-tree, where $a \geq 2$ and $b \geq 2a-1$, the total number of node splittings at height h is at most $n/\lfloor (b+1)/2 \rfloor^h$.*

Proof. The first node created at height h is created by a node splitting at height $h-1$. Each node splitting at height h increases the number of nodes at height h by one. By a simple induction it follows that after the first node splitting at height h all nodes at height h have at least $\lfloor (b+1)/2 \rfloor$ children. Since there are at most $n/\lfloor (b+1)/2 \rfloor^h$ nodes at height h in a tree with n leaves and height at least $h+1$, the theorem follows. □

Corollary 1 *In a sequence of n insertions on an initial empty (a,b)-tree, where $a \geq 2$ and $b \geq 2a-1$, the total number of node splittings is at most $4n/b$.*

Theorem 3 *In a sequence of i insertions and d deletions on an initial empty (a,b)-tree, where $a \geq 2$ and $b \geq 2a$, the total number of node splittings and node fusions at height h in the tree is $O(\delta^h(i+d))$, where $\delta < 1$ is a constant depending on a and b.*

Proof. Let $\alpha = \min\{2a-1, \lceil (b+1)/2 \rceil\}$ and $\beta = \max\{2a-1, \lceil (b+1)/2 \rceil\}$. Furthermore let $\delta_1 = 1/(2\alpha - 2a + 1)$, $\delta_2 = (1+\delta_1)/(b+1-\beta)$, and $\delta = \max\{\delta_1, \delta_2\}$. Note that since $a \geq 2$ and $b \geq 2a$, it follows that $a+1 \leq \alpha \leq \beta \leq b-1$, $\delta_1 \leq 1/3$, $\delta_2 \leq 2/3$ and $\delta \leq 2/3$

For each non-root node v we define a potential $\phi(v)$ given by the graph in Figure 4. For the root we define $\phi(v)$ similarly, except that the potential is also zero when the root has degree less than α. Finally we let $\phi_h = \sum_{v\,:\,\mathrm{height}(v)=h} \phi(v)$. Intuitively, ϕ_h is a measure of the unbalancedness at height h.

Let SP_h and FS_h denote respectively the number of node splittings and node fusions performed at height h, and $SP_0 = i$ and $FS_0 = d$. By induction on the length of the sequence of operations we will show that the following invariant holds for $h \geq 1$:

$$\phi_h \leq \delta(SP_{h-1} + FS_{h-1}) - (SP_h + FS_h) \ . \tag{1}$$

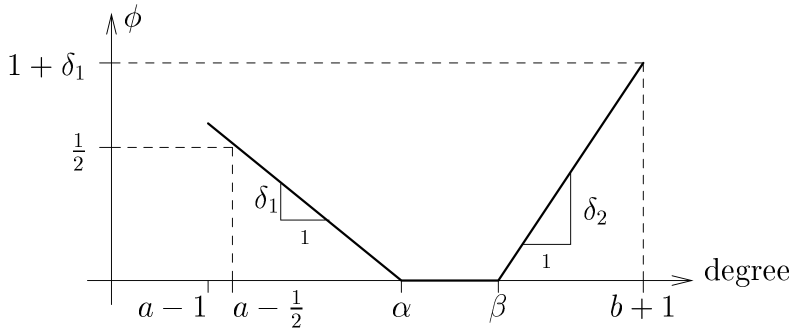

Figure 4: The potential $\phi(v)$ of a node.

By definition $\phi(v) \geq 0$ and therefore $\phi_h \geq 0$. From (1) we then immediately have $SP_h + FS_h \leq \delta(SP_{h-1} + FS_{h-1})$, i.e. $SP_h + FS_h \leq \delta^h(SP_0 + FS_0) = \delta^h(i + d)$.

We now turn to the proof of (1). Initially $\phi_h = SP_h = FS_h = 0$ for all $h \geq 0$, and (1) holds.

Next consider the case where SP_{h-1} has been increased because a leaf has been inserted ($h-1=0$) or there has been a node splitting at height $h-1$ ($h-1>0$). Then $\Delta\phi_h \leq \delta_2$ and the r.h.s. of (1) increases by δ. Since $\delta \geq \delta_2$, (1) remains valid. If a node with height h should be split because it gets degree $b+1$, then the splitting creates two nodes v_1 and v_2 with degree $\lfloor(b+1)/2\rfloor$ and $\lceil(b+1)/2\rceil$, where $\phi(v_1) \leq \delta_1$ and $\phi(v_2) = 0$ (since $\alpha - 1 \leq \lfloor(b+1)/2\rfloor \leq \beta$ and $\alpha \leq \lceil(b+1)/2\rceil \leq \beta$). It follows that $\Delta\phi_h \leq -(1+\delta_1) + \delta_1 = -1$ and the r.h.s. of (1) decreases by one, i.e. (1) remains valid.

If FS_{h-1} has been increased because a leaf has been deleted ($h-1=0$) or there has been a node fusion at height $h-1$, then $\Delta\phi_h \leq \delta_1$ and the r.h.s. of (1) increases by δ. Since $\delta \geq \delta_1$, (1) remains valid. A node fusion at height h fusions two nodes with degree respectively $a-1$ and a and with potential respectively $1/2 + \delta_1/2$ and $1/2 - \delta_1/2$, i.e. $\Delta\phi_h = -1$. Since the r.h.s. of (1) also decreases by one, (1) remains valid. For a node sharing at height h we have $-(\delta_1 + \delta_2) \leq \Delta\phi_h \leq 0$ and (1) remains valid since the r.h.s. of (1) does not change. □

Corollary 2 *In a sequence of i insertions and d deletions on an initial empty (a,b)-tree, where $a \geq 2$ and $b \geq 2a$, the total number of node splittings and node fusions is $O(i+d)$. If $b \geq (2+\epsilon)a$, for some $\epsilon > 0$, the number of node splittings and node fusions is $O(\frac{1}{\epsilon a}(i+d))$*

Proof. Follows from Theorem 3, the definition of δ, and $\sum_{i=1}^{\infty} \delta^i = \frac{\delta}{1-\delta}$ for $|\delta| < 1$. □

The number of node sharings at height h is bounded by the number of node fusions or leaf deletions at height $h-1$.

References

[1] Michael T. Goodrich and Roberto Tamassia. *Data Structures and Algorithms in JAVA*. John Wiley & Sons, Inc, 1998.

[2] Kurt Mehlhorn. *Data Structures and Algorithms 1: Sorting and Searching*. Springer Verlag, Berlin, 1984.