STRATIFIED BALANCED SEARCH TREES

J. van Leeuwen and M.H. Overmars

RUU-CS-81-4

February 1981

Rijksuniversiteit Utrecht

Vakgroep informatica

Princetonplein 5
Postbus 80.002
3508 TA Utrecht
Telefoon 030–53 1454
The Netherlands

STRATIFIED BALANCED SEARCH TREES

J. van Leeuwen and M.H. Overmars

Technical Report RUU-CS-81-4

February 1981

Department of Computer Science
University of Utrecht
P.O. Box 80.002
3508 TA Utrecht, the Netherlands

STRATIFIED BALANCED SEARCH TREES*

Jan van Leeuwen and Mark H. Overmars**

Department of Computer Science, University of Utrecht
P.O. Box 80.002, 3508 TA Utrecht, the Netherlands

Abstract. We develop a new perspective on trees, that enables us to distinguish and analyse many different subclasses of known classes of (height-)balanced search trees in a uniform manner. The approach shows that a great many different local constraints, including an arbitrary degree of density, can be enforced on everyday balanced search tree models, without losing the O(log n) bound on the time for insertions, deletions and finds. The theory extends known concepts from the study of B-trees.

1. Introduction

Search trees (cf. Knuth [6]) are used to structure tabular information for efficient retrieval. Assuming the degree of nodes is bounded, search trees are normally called "balanced" if the maximum path-length from the root to any leaf is less than c.log n, where n is the current number of tabular items ("elements") and c a constant depending on the type of search tree only. Ever since AVL-trees were discovered in 1962 (Adel'son-Vel'skii & Landis [1], cf. [6]), a fair number of different criteria of balance have been proposed, that could be maintained in O(log n) steps when single elements were inserted or deleted. Typically, balance is maintained by enforcing and maintaining a suitable condition on the height or the weight of the subtrees at each individual node or, when all search-paths are kept equally long, by allowing a degree of freedom in the branching at every

---

*A preliminary version of this paper was presented at the meeting on "Effiziente Algorithmen und Datenstrukturen", Oberwolfach, Feb. 2-6, 1981.

**The work of this author is supported by the Netherlands Organization for the Advancement of Pure Research (ZWO).

node. Among the many remarkable types of balanced search trees that have been identified, the following are of interest to us:

(i) AVL-trees
(ii) generalized AVL-trees
(iii) one-sided height-balanced trees
(iv) one-sided k-height balanced trees
(v) power trees
(vi) 2-3 trees
(vii) B-trees
(viii) symmetric binary B-trees
(ix) son-trees
(x) brother trees
(xi) 1-2 brother trees
(xii) right brother trees
(xiii) k-right brother trees
(xiv) 2-3 brother trees
(xv) height-balanced 2-3 trees
(xvi) neighbour trees
(xvii) k-neighbour trees
(xviii) αBB-trees
(xix) BB[α]-trees

Readers interested in a precise definition of these classes are referred to the open literature or to e.g. Olivié [8]. No detailed knowledge of any type of trees except (i), (vi) and (vii) (see Knuth [6]) is required for an understanding of this paper.

The many known classes of balanced trees are often very different and hard to relate to one another, yet their maintenance algorithms all seem to rely on a very limited number of different techniques. One might hope, then, that out of all these different experiments in balanced tree design some sort of unifying theory could be distilled, to treat many classes in an integral manner. In this paper we propose such a theory for discussing and analysing many different subclasses (of further constrained trees) of virtually all classes of balanced trees listed above in one single framework.

An interesting attempt at providing a uniform theory for the implementation and study of balanced trees was presented by Guibas & Sedgewick [5] in their "dichromatic" framework. They consider binary trees in which every node is allowed to carry one bit (its "color", say, red or black) to store balance information. They characterize a small number of local balancing transformations and argue that various known classes of trees, including AVL- and B-trees, and their maintenance algorithms can be embedded in the dichromatic framework by enforcing conditions on the occurrences of red and black nodes along the search paths. While the desired generality is achieved, the embeddings are not very straightforward and do not preserve the own identity that classes of balanced trees often have in an easily recognizable form.

We shall develop a different perspective on balanced trees, by abstracting a number of common features of "height-balanced" trees related to the locality of the balancing criteria and enforcing it as conditions that all trees must satisfy. Strictly speaking, the conditions enforce a certain regularity, called stratification, that need not be present in all trees of a given type but that identifies a subclass of that type. We prove that all stratified trees (of a certain type X) are balanced and can be maintained (as stratified trees of the same type X) by means of one master update algorithm in O(log n) steps whenever elements are inserted or deleted.

Once the formalism is explained (sections 2 and 3), it will appear that stratification basically extends the idea of B-trees to a higher conceptual level. The approach will show that a great many different, local constraints can be enforced on everyday balanced search tree models without losing the O(log n) bound on the time for insertions, deletions and finds. An intriguing, but direct consequence of our theory will be, for example, that for each $\varepsilon > 0$ one can distinguish a subclass of AVL-trees in which the proportion of not perfectly balanced nodes is less than ε, while trees in this class can nevertheless be maintained in O(log n) steps.

Most applications (section 5) concern the issue of density. For trees density normally refers to the minimal number of elements that will be packed in a tree of given height. The resultant idea of packing nodes to highest degree was exercised in the class of "dense multiway trees" proposed by Culik, Ottmann and Wood [4]. While succesful in updating these trees in O(log n) steps under insertions, they report an intuition that routines for deleting elements may need to be more complex if a sufficient degree of density is to be maintained. The theory we present will show that virtually every type of height-balanced trees can be constrained to an arbitrary degree of density, while both insertions and deletions can still be processed in O(log n) steps at a time.

2. Proper classes of trees and varieties.

While it is not strictly necessary for our theory, we shall adapt the convention that data elements are stored only at the leaves of a search tree. We shall not explicitly distinguish between search trees (with stored data) and their underlying graphical structure, when we discuss classes of search trees and their properties below. Let us assume that some class X of (balanced) trees is given. It is not necessary that an efficient updating algorithm is given with it, although we do assume that X-trees are meant for use in a dynamic environment. For each k, let X_k be the subclass of trees in X of height k. We shall allow that trees of small height (for small sets) are a bit irregular, but we do want X to behave well for larger size sets.

Definition. X is α-proper if and only if for each $t \geq \alpha$ there is a tree in X with t leaves.

If X is α-proper, then each set of size $\geq \alpha$ can be accommodated by an X-tree. Virtually all known classes of balanced trees are 1-proper, hence α-proper for any $\alpha \geq 1$.

Disturbances of balance in a tree are normally resolved by suitably restructuring subtrees of some bounded size along a search path. It implies that there is some notion of a "good" subtree. Let Z be a set of trees of the same height β. Let l_Z and h_Z be the smallest and largest number of leaves, respectively, that members of Z can have. Note that we do not require that $Z \subseteq X_\beta$.

Definition. Z is a β-variety if and only if the following conditions are satisfied:

(i) all trees in Z have height β,

(ii) $1 < l_Z < h_Z$,

(iii) for each t with $l_Z \leq t \leq h_Z$ there is a tree in Z with exactly t leaves.

It should be clear that β-varieties are very easy to construct. For many known classes of balanced trees X is X_β a β-variety, for every β greater than or equal to 1 or 2.

Given a β-variety Z and a α-proper class of trees X, we would like to express that the trees of Z can figure as "good" subtrees in trees of X.

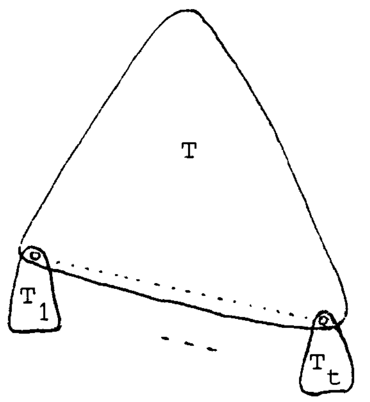


Figure 1.

<u>Notation</u>. Let $T_1, \ldots, T_t$ be trees and let T be a tree with leaves $x_1, \ldots, x_t$ from left to right. By $T[T_1, \ldots, T_t]$ we shall denote the tree obtained from T by grafting each T_i onto x_i $(1 \le i \le t)$. See figure 1.

<u>Definition</u>. Z is a regular β-variety for X if and only if the following conditions are satisfied:

(i) Z is a β-variety,

(ii) for each $t \ge \alpha$ and $T \in X$ with t leaves and all $T_1, \ldots, T_t \in Z$ is $T[T_1, \ldots, T_t] \in X$.

If $T \in X_k$, then in (ii) $T[T_1, \ldots, T_t]$ will be in $X_{k+\beta}$.

<u>Example</u>. Let X be the class of AVL-trees (cf. [6]). X is α-proper for every $\alpha \ge 1$. For each $\beta \ge 2$ is X_β a regular β-variety for X, considered as a 1-proper class.

<u>Example</u>. Let X be the class of trees with a root of degree d (d fixed) and all other internal nodes of degree 2. Let Z be the set of binary trees of height 2. X is d-proper and Z is a regular 2-variety for X.

For many classes of balanced trees X listed in Section 1 (but nót for e.g. B-trees and BB[α]-trees) is X_β a regular β-variety of X for all β larger than 1 or 2.

3. Stratification

Let X be an α-proper class of trees. Assume there is a regular β-variety Z for X. We will show how to obtain a subclass of very special trees in X, essentially by "layering" trees from Z. Let $K = \max\{\alpha l_Z, \left\lceil \frac{l_Z - 1}{h_Z - l_Z} \right\rceil l_Z\} - 1$ and let γ be the smallest integer such that for each t with $\alpha \le t \le K$ there is a $T \in X$ of height $\le \gamma$ with exactly t leaves. (γ exists.)

Definition. The class of Z-stratified trees (in X) is the smallest class of trees satisfying the following properties:

(property I) each $T \in X$ of height $\le \gamma$ for which the number of leaves t satisfies $\alpha \le t \le K$ is Z-stratified,

(property II) if T is Z-stratified and has t leaves and $T_1, \ldots, T_t \in Z$, then $T[T_1, \ldots, T_t]$ is Z-stratified.

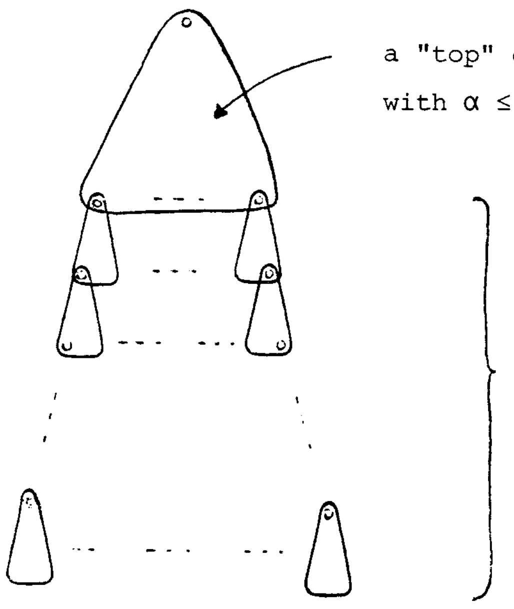
Notation. The class of Z-stratified trees (in X) will be denoted as S(X, Z).

Lemma 3.1. $S(X, Z) \subseteq X$.

Proof

By induction on the definition of Z-stratified trees. Note that each Z-stratified tree has $\ge \alpha$ leaves. Applying property II to any tree in X with $\ge \alpha$ leaves keeps the result in X, by the definition of Z. □

It should be clear that $T \in S(X, Z)$ if and only if T can be decomposed into the following form:


a "top" of height $\le \gamma$, with a number of leaves t with $\alpha \le t \le K$ (as defined)
s layers, some $s \ge 0$

(3.2.)

with each of the component trees in the layers chosen from Z. Each layer is obtained by yet another application of property II.

Lemma 3.3. S(X, Z) is α-proper.

Proof

Consider Z-stratified trees as they are decomposed into layers. We shall prove the following claim by induction on s:

Claim: $\alpha \le t \le Kh_Z^s \Leftrightarrow$ there is a $T \in S(X, Z)$ with t leaves and $\le s$ layers.

The $\Leftarrow$ -part is obvious.

The $\Rightarrow$ -part is immediate for $s = 0$. Let the claim be true for s. Consider any t with $\alpha \le t \le Kh_Z^{s+1}$. Since all t with $\alpha \le t \le Kh_Z^s$ are covered by the induction hypothesis, we only need to consider t with $Kh_Z^s + 1 \le t \le$ $\le Kh_Z^{s+1}$. By induction we know that for each y with $\alpha \le y \le Kh_Z^s$ there is a $T \in S(X, Z)$ with y leaves and at most s layers. We shall argue that the range left for t can be covered by adding one more layer to these trees.

Adding a layer to a tree with y leaves yields trees that can have any number of leaves t with $yl_Z \le t \le yh_Z$. In this way one can cover all intervals $[yl_Z, yh_Z]$ for t, for y ranging from α to Kh_Z^s. If $(y+1)l_Z \le yh_Z + 1$, then the interval for y interlaces with the next one. Only when $(y+1)l_Z > yh_Z + 1$, i.e., when $y < \frac{l_Z-1}{h_Z-l_Z}$, will there be a gap in between. We note that in this case

$$(y+1)l_Z - 1 \le (\left\lceil \frac{l_Z-1}{h_Z-l_Z} \right\rceil - 1 + 1) - 1 \le K \le Kh_Z^s$$

and, hence, the t-values in the gap are contained already in the range covered by the induction hypothesis. Observe next that the first interval (with $y = \alpha$) starts exactly at the beginning of the range for t or earlier, as $\alpha l_Z \le Kh_Z^s + 1$. (For $s = 0$ it follows by the choice of K, for $s \ge 1$ by a growth argument.) The last interval (with $y = Kh_Z^s$) ends exactly at Kh_Z^{s+1}, the end of the range of t under consideration. It follows that the joint intervals cover all t with $Kh_Z^{s+1} + 1 \le t \le Kh_Z^s$ and, hence, that all t-values in this range can appear as the number of leaves of some Z-stratified tree with at most s + 1 layers. □

We conclude that in order to represent sets of size $\ge \alpha$ we can replace X by its subclass S(X, Z). It remains to be seen whether S(X, Z) exhibits a comparable efficiency. We consider searches in Z-stratified trees.

Theorem 3.4. Any Z-stratified tree with n leaves has height O(log n).

Proof

Let $T \in S(X, Z)$ have n leaves and s layers. It follows that $\alpha l_Z^s \leq n$, hence $s \leq (\log n - \log \alpha)/\log l_Z$. The height of T is bounded by $\gamma + s.\beta$, thus by $\frac{\beta}{\log l_Z}.\log n + c$ for some constant c. □

It follows that representing sets by trees in S(X, Z) is acceptable as far as the resulting complexity of searches is concerned. We consider the possibility of dynamically maintaining Z-stratified trees in the next section. We note that no maintenance algorithm may actually be known for the class X itself.

It is useful to observe at this stage that there is a resemblance between Z-stratified trees and B-trees. Considering representation (3.2.), one might "collapse" the distinguished subtrees of any $T \in S(X, Z)$ into single nodes and obtain a multiway tree in which all internal nodes except possibly the root have degree d with $l_Z \leq d \leq h_Z$. In B-trees (Bayer & McCreight [2], also [6]) it is normally required that $h_Z \geq 2l_Z - 1$, but we make no such assumption here. The maintenance algorithms for Z-stratified trees as presented in the next section combine and extend the techniques used for updating B- and B*-trees (cf. Knuth [6]), as was done to some extent also in the study of "dense" multiway trees by Culik, Ottmann and Wood [4]. Our present theory may show that multiway trees are much more fundamental to the study of arbitrary balanced trees than has been noted until now and that ordinary B-trees are only the simplest instance of an entire family of classes of trees, tuneable to performance.

4. Maintenance of stratified trees.

Let X be α-proper, Z a regular β-variety for X and S(X, Z) as defined in the previous section. We shall assume that Z-stratified trees are presented in storage in such a manner that a decomposition as in (3.2.) is at hand. This causes no difficulty, because all building blocks (the top and the subtrees from Z) are of bounded size and can be delineated by suitable markings.

Theorem 4.1. Insertions in Z-stratified trees can be processed in O(log n) steps.

Proof

Let $T \in S(X, Z)$ and suppose we must insert a new data-element D.

If T currently stores $t \leq K$ items, then we insert D "manually" at the proper place among the leaves and rebuild T as a Z-stratified tree on $t + 1$ leaves. By lemma 3.3. this can be done and leads to a tree with at most one layer. The amount of work required is O(K), hence O(1).

Let T currently have more than K items. Thus T will consist of a "top" T_0 of the necessary specifications and $s \geq 1$ layers below it. Search with D down T to find where D must be inserted among the leaves. Let the blocks passed by the search path be $T_0, T_1, \ldots, T_s$, where T_1 to T_s are trees from Z. If T_s has t leaves and $t < h_Z$, then it is sufficient to place D in the right order among the leaves and to rebuild T_s as a Z-tree on $t + 1$ leaves. (This can always be done and keeps T stratified.) If $t = h_Z$, then we have to do more work and may be forced even to "split off" a new Z-tree in the current layer which, consequently, must be inserted in T_{s-1}. And this can propagate through several more layers upwards. Very generally, let us consider the insertion of a "leaf" in T_i for $i > 1$. Let T_i currently have t leaves.

If $t < h_Z$, then we can rebuild T_i as a Z-tree on $t + 1$ leaves and are done.

If $t = h_Z$, then we shall examine the "brothers" of T_i to see if we can move elements over and make room for the element to be inserted. Note (see figure 2) that T_i is a "leaf" of T_{i-1} and that there are at least

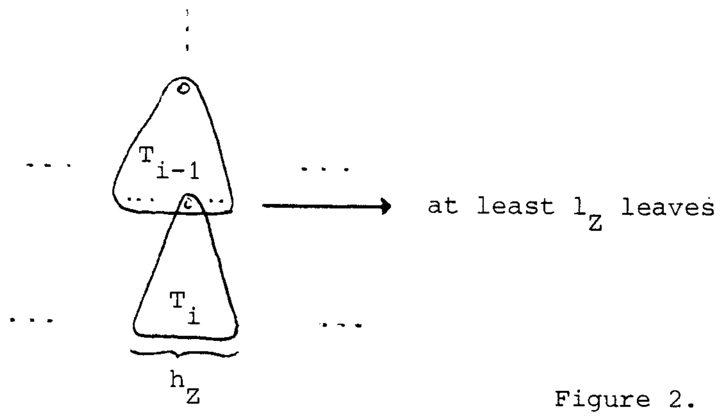


Figure 2.

$l_Z - 1$ neighboring leaves (Z-trees). If one of them still has room, i.e., less than h_Z leaves itself, then we can shift elements over and redistribute them over l_Z subtrees such that <u>with the new element included</u> no subtree needs to have more than h_Z leaves. It requires the reconstruction of up to l_Z subtrees as Z-trees (and a revision of the search queries at their nodes), but is still O(1) steps of work.

If all of these $l_Z - 1$ neighboring brothers are full, i.e., have h_Z leaves, then consider the entire row of $(l_Z - 1)h_Z + h_Z + 1$ elements we must accomodate. Clearly l_Z subtrees are not sufficient, but $l_Z + 1$ are, because of the following inequality:

$$(l_Z + 1)l_Z = l_Z h_Z - l_Z(h_Z - l_Z - 1) < l_Z h_Z + 1 < l_Z h_Z + h_Z = (l_Z + 1)h_Z$$

It easily follows that the $l_Z h_Z + 1$ elements (roots of subtrees) can be put together in $l_Z + 1$ Z-trees in this layer, one more Z-tree than we had. Thus we succeed, provided we carry out an insertion in T_{i-1}.

So the procedure repeats, until it eventually gets to T_1 (if it didn't halt before that). For an insertion into T_1 we can not follow the same procedure, because T_0 is not a Z-tree. (In particular, T_0 need not provide $l_Z - 1$ brothers.) Consider T_0 and the entire first layer (see figure 3). Let the first layer have a total of t leaves, hence $t \le Kh_Z$. Insert the

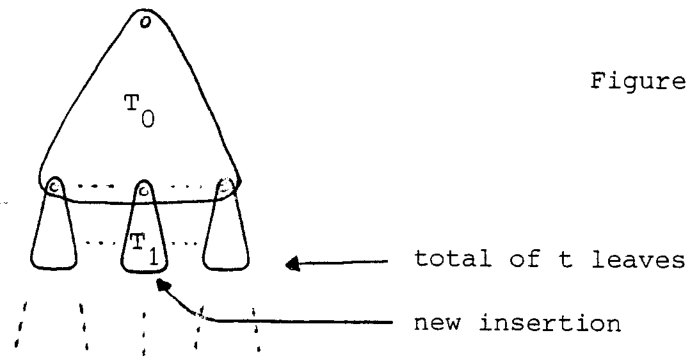


Figure 3.

new node at the proper place and rebuild the entire portion of the tree as a Z-stratified tree on t + 1 elements. By lemma 3.3. one can do so, with a resulting tree of at most 2 layers. The amount of work required is $O(Kh_Z)$, thus O(1). Appending the lower layers (automatically as the subtrees of the t + 1 elements accomodated for) maintains the conditions of a Z-stratified tree.

The total amount of work adds up to O(s), which is O(log n). □

It is noted that the insertion in the top of the tree could have been dealt with more easily, had we assumed that $\alpha \ge l_Z$. This not being the case, a rather massive reconstruction is required to carry the proof through.

Theorem 4.2. Provided the total number of elements remains $\ge \alpha$, deletions in Z-stratified trees can be processed in O(log n) steps.

Proof

Let $T \in S(X, Z)$ and suppose we must delete an element D.

If T currently has $t \le K$ elements, then we just delete D "manually" and rebuild T as a Z-stratified tree with $t - 1$ elements (provided $t - 1 \ge \alpha$). This can be done and requires no more than $O(K)$ steps of work.

Let T currently have more than K items. Thus T will consist of a top T_0 and $s \ge 1$ layers attached to it. Search with D down T to find where it is located among the leaves. Let the blocks passed on the way down be $T_0, T_1, \ldots, T_s$, where T_1 to T_s are Z-trees. As for insertions, deletions can propagate upwards. We shall consider the deletion of a leaf node from T_i for $i > 1$. Assume that T_i currently has t leaves.

If $t > l_Z$, then we can perform the deletion and rebuild T_i as a Z-tree on $t - 1$ leaves. It requires $O(1)$ steps of work and we are done.

If $t = l_Z$, then we shall examine the "brothers" of T_i (as Z-trees) to see if one of them has an element to spare, i.e., if one has $> l_Z$ elements. Note that T_i has (at least) $l_Z - 1$ brothers (as in the proof of 4.1.) and if one of them has $> l_Z$ elements, then we can shift over and redistribute elements so that after the desired deletion has been performed all of these l_Z subtrees still have $\ge l_Z$ leaves each. It requires that up to l_Z subtrees are reconstructed (as Z-trees), but this takes only $O(1)$ steps of work.

If all of the $l_Z - 1$ neighboring brothers are "minimally filled", i.e., have l_Z elements, then consider the entire row of $(l_Z - 1)l_Z + l_Z - 1$ elements that must be accomodated. Clearly they do not fit into l_Z Z-trees anymore, but they do in $l_Z - 1$, as the following inequality lets us conclude:

$$(l_Z - 1)l_Z < l_Z^2 - 1 \le l_Z^2 + l_Z(h_Z - l_Z) - h_Z = (l_Z - 1)h_Z$$

Construction of $l_Z - 1$ Z-trees takes again $O(1)$ steps of work, but note that it gives us one component less than the number we had. Thus to succeed, we must continue and carry out a deletion on T_{i-1}.

So the procedure repeats, until eventually it gets to T_1 (if it didn't finish before). If the first layer has a total of t leaves then do the necessary deletion in T_1 and, like we did in the proof of 4.1., rebuild T_0 and the entire first layer as a Z-stratified tree on $t - 1$ leaves. (Note that $t \ge \alpha l_Z > \alpha$, which shows that the reconstruction can be carried out.)

The total amount of work is again $O(s)$, which is $O(\log n)$. □

In the proofs of 4.1. and 4.2. the details of how to adapt the assignment of search queries at the nodes have been omitted. The changes are all local and left as an easy exercise to the reader.

The maintenance routines for stratified trees prove the results we were after, but are not necessarily practical. For specific classes $S(X, Z)$ one may wish to inspect fewer brothers of the components and use a simpler procedure at the top of the tree.

5. Applications

We shall apply the idea of stratification to distinguish some remarkable subclasses of common classes of balanced trees.

Terminology. A class of search trees is said to be log n-maintainable if and only if there is some constant c such that insertions, deletions and finds can be performed within c log n steps on any tree with n leaves in the class (n large enough).

The results of Sections 3 and 4 can be summarized into the following statement.

Theorem 5.1. Let X be α-proper and Z a regular β-variety for X. Then $S(X, Z)$ is an α-proper and log n-maintainable subclass of X.

It follows that in order to distinguish interesting subclasses of a given class X (which need not be log n-maintainable itself), it suffices to find suitable regular varieties for X.

AVL-trees

Let X be the class of AVL-trees. We know that X is 1-proper, thus we can take $\alpha = 1$. Observe that every X_β $(\beta \geq 2)$ is a β-variety. The following observation is crucial:

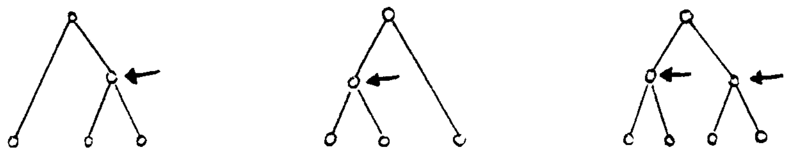
Lemma 5.2. If Z is a β-variety of AVL-trees, then Z is a regular β-variety for X.

We can immediately use the lemma to stratify with e.g. X_2, to obtain the following class of trees. Let a node be at level j if and only if the longest path from the node to a leaf has j edges.

Proposition 5.3. There exists a log n-maintainable class of AVL-trees in which every odd-numbered level, except perhaps the root-level, consists only of nodes that are in perfect balance.

Proof

Consider the distinct members of X_2 (see figure 4). Clearly X_2 is a 2-variety and hence, by lemma 5.2., a regular 2-variety for the class of AVL-trees (X). Take $Z = X_2$ and consider S(X, Z). The odd-numbered levels of Z-stratified trees, except perhaps those at the top, precisely

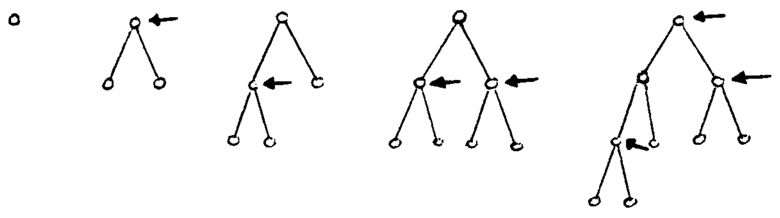


($\leftarrow$ denotes odd-level nodes)

(a) (b) (c)

Figure 4.

contain the nodes of the middle level (pointed at by the $\leftarrow$ in figure 4) of each component X_2-tree. It is easily seen that they are in perfect balance the way they occur in the trees. At the top, just note that $K = \max\{1.3\ ,\ \left\lceil\frac{3-1}{4-3}\right\rceil 3\} - 1 = 5$. Thus the top-portions of Z-stratified trees must include the trees displayed in figure 5 (only non-isomorphic copies are shown). All nodes, except the root in case (e), that will



($\leftarrow$ denotes odd-level nodes)

(a) (b) (c) (d) (e)

Figure 5.

occur in odd-numbered levels, will be perfectly balanced. Thus S(X, Z) is the class as desired. □

Theorem 5.4. For each $\varepsilon > 0$, there is a log n-maintainable class of AVL-trees in which the proportion of nodes that are not in perfect balance is less than ε (provided the number of leaves is sufficiently large).

Proof

Determine k such that $\frac{1}{2^k-2} < \varepsilon$. Consider the set Z consisting of the perfectly balanced trees on $2^k - 1$ and 2^k leaves, respectively, as displayed in figure 6. While Z consists of only 2 trees, it is a valid

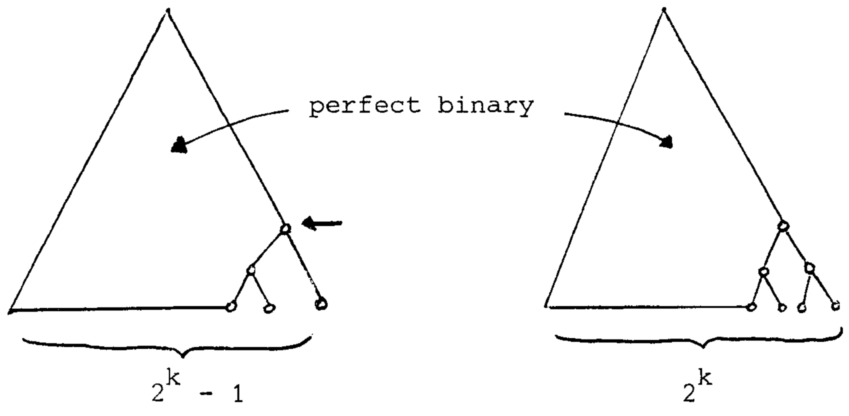


Figure 6.

k-variety, hence a regular k-variety for the class of AVL-trees. Consider trees in S(X, Z). If its number of leaves is large enough, then the top of a Z-stratified tree is small compared to the size of the layers. In each component of a layer at most one (the node pointed at by ← in figure 6) of at least $2^k - 2$ internal nodes can be out of balance. Thus the proportion of nodes that are not in perfect balance is $\leq \frac{1}{2^k-2} < \varepsilon$, provided the trees are large enough. Thus S(X, Z) is a class as desired. □

In the past there has been some attention for one-sided AVL-trees ([7], [9], [10]), i.e., AVL-trees in which the balance factors are only 0 or 1 (but never - 1). It should be clear that proposition 5.2. goes through for one-sided AVL-trees, once we eliminate trees of type (a) in figure 4 from the variety and force all trees in the top to be of an exact type as displayed in figure 5. Even theorem 5.4. goes through, because the simple variety used in the proof actually consists of one-sided AVL-trees Note that the maintenance algorithms for the classes referred to in proposition 5.3. and theorem 5.4. are in no way related to the "known" maintenance algorithms for AVL-trees in general.

Other classes of binary, ternary etc. trees.

The k-variety used in the proof of theorem 5.4. (cf. figure 6) or the obvious variant with higher degree nodes, is a regular variety for almost any class of balanced trees. Thus, stratification by means of such a variety will show that, theoretically, almost every type of balanced trees can be "packed" or "almost perfectly balanced", without losing the log n-maintainability of the class. With theorem 5.1. there should be no magic to density results. Clearly, the maintenance algorithms for the classes of dense trees construed may be worse on the average than for the unconstrained class, but this is the price to pay for density. Precise trade-offs may be an interesting subject for further study.

Symmetric binary B-trees.

Symmetric binary B-trees (SBB-trees) were introduced by Bayer [2] to obtain a suitable "binarization" of arbitrary B-trees. Olivié [8] noted a nice relationship between SBB-trees and a special class of 1-2 trees.

Definition. A 1-2 tree T is called a standard son tree if and only if the following conditions are satisfied for T (in addition to being a 1-2 tree):

(i) the depth of T is even,

(ii) the even levels of T contain no unary nodes.

It follows in particular that in standard son trees no unary node can have a unary son. Olivié [8] proved:

Theorem 5.5. There is a natural, one-to-one correspondence between SBB-trees and standard son trees.

Considering standard son trees more precisely, let X be the class of 1-2 trees and Z be the 2-variety of standard son trees of depth 2 (see figure 7). It is easily seen that Z is a regular 2-variety for X, with

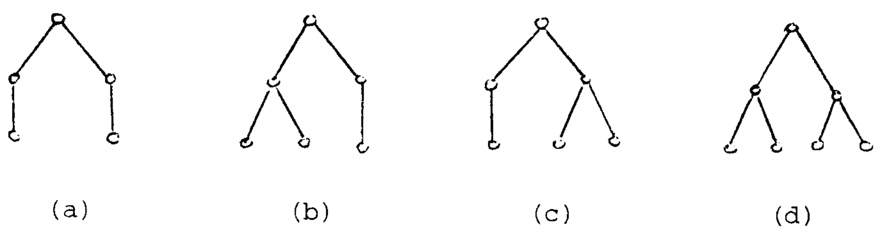


Figure 7.

$l_Z = 2$, $h_Z = 4$, $\alpha = 1$ and $K = 1$.

Theorem 5.6. With X and Z as defined, S(X, Z) is precisely the class of standard son trees.

From 5.1. one can immediately derive that the standard son trees are a log n-maintainable class. (The resulting algorithm proves to be very similar to Olivié's [8].) In fact, the result indicates very clearly that standard son trees, hence SBB-trees, are 2-4 trees in disguise.

B-trees

B-trees were originally introduces by Bayer and McCreight [3], but later extended in several ways (see e.g. Knuth [6], sect 6.2.4.). Essentially, a B-tree of order m is a tree which satisfies the following properties:

(i) all leaves have equal depth,
(ii) the root has a degree d, satisfying $2 \le d \le 2\left\lceil \frac{m}{2} \right\rceil - 1$,
(iii) all remaining nodes have a degree d satisfying $\left\lceil \frac{m}{2} \right\rceil \le d \le m$.

(We ignore the details of how "keys" are stored. Note that property (ii) gives a slightly sharper bound on the degree of the root than is usually stated.) The following result shows that stratified trees and B-trees are intimately related. Let X be the class of B-trees of order m, Z the 1-variety of trees with a root of degree d with $\left\lceil \frac{m}{2} \right\rceil \le d \le m$. Z is a regular 1-variety for X and one easily verifies the following result:

Theorem 5.7. With X and Z as defined, S(X, Z) is precisely the class of B-trees of order m.

From 5.1. the log n-maintainability of the class of B-trees is confirmed.

Interpreting nodes as tracks on a disk and m as the maximum number of records that fit on one track, B-trees are of use for practical file design in which nodes (tracks) are always filled to at least half the maximum capacity, with the possible exception of the root. Knuth [6] (p. 478) describes a variant type of B-trees, called B*-trees, in which all internal nodes except the root have a degree d satisfying $\left\lceil \frac{2m-1}{3} \right\rceil \le d \le m$. Thus, B*-trees guarantee a 67% minimum space utilization on every track (except for the root). Let X be the class of B*-trees of order m and Z the 1-variety of trees with a root of degree d satisfying $\left\lceil \frac{2m-1}{3} \right\rceil \le d \le m$. Z is a regular 1-variety for X and one easily verifies that $\alpha = 2$ and that K (the degree bound at the root) must equal the following value:

$$K = \max \{2\left\lceil\frac{2m-1}{3}\right\rceil, \left\lceil\frac{\left\lceil\frac{2m-1}{3}\right\rceil - 1}{m - \left\lceil\frac{2m-1}{3}\right\rceil}\right\rceil \cdot \left\lceil\frac{2m-1}{3}\right\rceil\} - 1 =$$

$$= 2\left\lceil\frac{2m-1}{3}\right\rceil - 1$$

$$= 2\left\lfloor\frac{2m-1}{3}\right\rfloor + 1$$

(Compare [6], p. 478)

Theorem 5.8. With X and Z as defined, S(X, Z) is precisely the class of B*-trees of order m.

To obtain a statement about the maximum space utilization attainable in theory by B-trees, it is useful to distinguish the following class of trees:

Definition. Given l, m with $1 < l < m$, a B-tree of order (l, m) is any tree which satisfies the following properties:

(i) all leaves have equal depth,

(ii) the root has a degree d satisfying $2 \le d \le \max \{2l, \left\lceil\frac{l-1}{m-l}\right\rceil l\} - 1$,

(iii) all other nodes have a degree d satisfying $l \le d \le m$.

The factual theory of B-trees culminates in the following theorem.

Theorem 5.9. The class of B-trees of order (l, m) is 2-proper and log n-maintainable, for every $1 < l < m$.

Proof

Let X be the class of B-trees of order (l, m) and Z the l-variety of trees having a root of degree d with $l \le d \le m$. Z is a regular l-variety for X and it is easily seen that S(X, Z) is precisely the class X itself. The theorem immediately follows from theorem 5.1. □

By choosing l close to m, a log n-maintainable class of B-trees is obtained with a space utilization of nearly 100%. One might even take $l = m - 1$, although this would force the root node to have a degree up to $m^2 - 3m + 1$ and gives abominable constants in the O(log n) time bounds for the insertion and deletion routines.

Dense multiway trees.

The preceding analysis has shown that, theoretically, B-trees can be made arbitrarily dense. One can go even further and stratify B-trees themselves.

Definition. An internal node of a B-tree is called saturated if it has a maximum degree.

It will be an easy exercise for the reader to apply theorem 5.1. and prove the following, typical result (compare theorem 5.3.):

Theorem 5.10. There exists a log n-maintainable class of B-trees of order (l, m) of which all even-numbered levels, except for a few near the top, contain only saturated nodes, for every $1 < l < m$.

One can continue along this line and prove, very similar to theorem 5.4., that for every $\varepsilon > 0$ and $1 < l < m$ there is a log n-maintainable class of B-trees of order (l, m) in which the proportion of unsaturated nodes in less than ε, provided the number of leaves is large enough. It is hard to conceive of more densely packed trees!

Culik, Ottmann and Wood [4] pursued a different approach to obtain dense trees. Considering multiway trees with node degrees from 1 up to m permitted, they introduced the following concept:

Definition. A multiway tree T is called r-dense (for some r with $1 \leq r \leq$ $\leq m - 1$) if and only if the following conditions are satisfied:

(i) all leaves have the same depth,

(ii) the root of T has a degree d satisfying $2 \leq d \leq m$,

(iii) each unsaturated node different from the root either has only saturated brother and at least one such brother or has at least r saturated brothers.

Culik et.al. [4] proved that insertions in 1-dense multiway trees ($r = 1$) can be processed in O(log n) steps. The general case ($r \geq 1$) and the problem of handling deletions also, were left unsolved. While we have no answer to these problems to offer here, it turns out to be fairly easy to find log n-maintainable subclasses of the classes of r-dense multiway trees for any $r \geq 1$.

Theorem 5.11. For every r with $1 \leq r \leq m - 1$, there exists a log n-maintainable class of r-dense multiway trees.

Proof

Let X be the class of r-dense multiway trees. It can be shown that X is 2-proper. Let Z be the 2-variety of trees which have a root of degree m and at least r of the (internal) nodes at depth 1 saturated as well. Z is a regular 2-variety of X. Hence S(X, Z) is a class of trees as desired. □

It should be clear that r-dense trees, even (m - 1)-dense trees, can be stratified further to obtain log n-maintainable subclasses of any arbitrary degree of packing. The results so obtained answer various questions from Culik et.al. [4] affirmatively about the existence of log n-maintainable classes of dense trees.

6. References.

(For references to the extensive literature on balanced trees see e.g. [8].)

[1] Adel'son - Vel'skii, G.M. and E.M. Landis, An information organisation algorithm, Doklady Akad. Nauk SSSR 146 (1962) 263-266, transl. Soviet·Math. Dokl. 3 (1962) 1259-1262.

[2] Bayer, R., Symmetric binary B-trees: data structure and maintenance algorithms, Acta Informatica 1 (1972) 290-306.

[3] Bayer, R. and E.M. McCreight, Organisation and maintenance of large ordered indexes, Acta Informatica 1 (1972) 173-189.

[4] Culik II, K., Th. Ottmann and D. Wood, Dense multiway trees, Bericht 77, Inst. f. Angew. Informatik u. form. Beschreibungsverfahren, Universität Karlsruhe, Karlsruhe, 1978.

[5] Guibas, L.J. and R. Sedgewick, A dichromatic framework for balanced trees, Proc. 19th Annual IEEE Symp. on Foundations of Computer Science, Ann. Arbor, Oct. 16-18 (1978), pp. 8-21.

[6] Knuth, D.E., The art of computer programming, vol. 3: sorting and searching, Addison-Wesley Publ. Comp., Reading, Mass, 1973.

[7] Kosaraju, S.R., Insertion and deletion in one-sided height-balanced trees, C. ACM 21 (1978) 226-227.

[8] Olivié, H., A study of balanced binary trees and balanced one-two trees, Ph.D. Thesis, Dept. of Mathematics, University of Antwerp (UIA), Antwerp, 1980.

[9] Ottmann, Th. and D. Wood, Deletion in one-sided height-balanced search trees, Int. J. Comput. Math. 6 (1978) 265-271.

[10] Zweben, S.H. and M.A. McDonald, An optimal method for deletion in one-sided height-balanced trees, C. ACM 21 (1978) 441-444.