Scan
"Preprint"
Add to A6265

Height Balance Distribution of Search Trees

Ricardo A. Baeza-Yates

Depto. de Ciencias de la Computación
Universidad de Chile
Casilla 2777, Santiago, Chile

Keywords: Analysis of algorithms, data structures, binary search trees, balanced search trees.

1 Introduction

Binary search trees are well-known data structures used to solve the dictionary problem. Local balancing in binary search trees is a heuristic applied to the fringe of a tree [Gon83, GR91, PM85]. This technique lies between no balance (binary search trees) and rigid balance disciplines (height [AVL62], weight [NR73], or internal path balance [Gon83]). The basic idea is to keep the fringe of a tree balanced using local rotations. The origin of this heuristic can be traced to the work of Bell [Bel65], and Walker and Wood [WW76].

In this paper we study the height balance distribution of binary search trees, with or without local/global balancing. In Section 2 we study binary search trees (BSTs). In Section 3 we extend this study to locally balanced search trees and we analyze a generalized heuristic for local balance in binary search trees. From this we obtain the expected number of comparisons in successful and unsuccessful searches and we also determine the optimal heuristic for this schema when uniform, random keys are used. We also propose an implementation which in the worst case uses only one pointer per key.

In Section 4 we study the balance distribution of AVL trees and a variant of them. We also include exact results for small AVL trees.

2 Binary Search Trees

Binary search trees are used to lexicographically store a set of keys [Knu73, GR91]. Every node of the tree stores a key, say k, and two pointers: the left and the right subtrees, such that any key in the left subtree is less or equal than k, and any key in the right subtree is greater than k. The height of a tree is defined as the number of edges in the longest path through the tree.

We define the *height balance factor* of a node to be

$$hb = |\ \text{height}(\textit{left subtree}) - \text{height}(\textit{right subtree})\ | \geq 0\ .$$

For example, in AVL trees the hb of a node is always 0 or 1. We define $\overline{hb}_k(n)$ as the expected number of nodes with $hb = k$ when the number of elements in the tree is n.

A lower bound in the expected number of nodes with $hb = k$ in a random binary search tree of n elements [BR82] is given by

$$\overline{hb}_k(n) = \frac{1}{n}\sum_{i=1}^{n}(\overline{hb}_k(i-1) + \overline{hb}_k(n-i)) = \frac{n+1}{n}\overline{hb}_k(n-1) \tag{1}$$

so that

$$\overline{hb}_k(n) = \frac{n+1}{j+1}\overline{hb}_k(j), \text{ for } n > j\ .$$

$\overline{hb}_k(j)$ for a small j has to be computed directly using the exact values up to $j-1$ and equation (1). Therefore, we have

$$\frac{\overline{hb}_k(n)}{n+1} \geq \frac{\overline{hb}_k(j)}{j+1}, \text{ for } n > j. \tag{2}$$

Table 1 shows the lower bounds of the fraction of the nodes with $hb = k$, a lower bound on the expected height balance factor (E(k)), the percentage of the nodes included and the size j of the subtrees considered. In all cases the values are obtained with an ad-hoc program that computes all the possible trees and transitions up to size $j-1$. The number of trees of size $j-1$ is also shown in Table 1.

We use the fact that all subtrees which are isomorphic upon symmetry have the same behavior with respect to insertions.

k	Binary Search Tree (lower bound)	Basic Heuristic (lower bound)	Partial AVL Trees (lower bound)	AVL Trees [BYGZ90] (lower bound)	(upper bound)
0	.3936707	.5734669	.648306	.5637	.7799
1	.2425519	.2825299	.311694	.2211	.4373
2	.1241809	.0464547			
3	.0667827	.0152199			
4	.0358048	.0043360			
5	.0184752	.0009358			
Lower bound in E(k)	1.013	.445	.280	.221	
% of nodes	89.47	92.31	96.00	75	75
Subtree size j	18	25	25		
Trees of size $j-1$	113310	80484	173024		

Table 1: $\frac{\overline{hb}_k(n)}{n+1}$ for $n > j$.

We see that at least 63.6% of all internal nodes in a binary search tree are roots of subtrees with AVL balance, and that

$$0.3936(n+1) \leq \overline{hb}_0(n) \leq 0.4989(n+1)\ .$$

3 Locally Balanced Search Trees

This heuristic has been rediscovered many times receiving different names by Itai and Rodeh ("modified binary search trees") [IR80], Bagchi and Reingold ("weakly balanced trees") [BR82], Huang and Wong ("iR trees") [HW83], and Greene ("diminished trees") [Gre83]. Further analysis was done by Poblete and Munro [PM85], Hermosilla and Olivos [HO85], and implicitly in Guibas and Sedgewick [GS78].

We define a *t-locally balanced tree* as a binary search tree where insertions falling in a subtree of size $2t$ produce a reorganization from which the median of the $2t+1$ elements is chosen as the root of two subtrees of size t. This definition coincides with that of Greene [Gre83] and Poblete and Munro [PM85], and when $t = 2^i - 1$, it also coincides with the iR trees [HW83]. The case $t = 0$ is a simple binary search tree and the case $t = 1$ is called the *basic heuristic* [BR82, IR80].

The expected unsuccessful search time in these trees [Gre83, PM85] is

$$C'_n = \frac{1}{H_{2t+2} - H_{t+1}} H_{n+1} + O(1) ,$$

where n is the number of elements in the tree and $H_i = \sum_{j=1}^{i} 1/j$ denotes the harmonic numbers.

3.1 A Generalized Heuristic

The generalized heuristic is defined as before, except that the root of the reorganization schema is not selected as the median. If the elements are $\{x_1, \ldots, x_{2t+1}\}$, the element x_j is chosen as the root with probability p_j such that $\sum_{j=1}^{2t+1} p_j = 1$. This probabilities may be fixed values or values selected dynamically. These values will be based on the probability distribution of the elements.

Intuitively, if we model the tree as being built from uniform, random keys, then the median is the best choice. We can prove this using the analysis technique of Poblete and Munro [PM85].

Let $A^j_{n,k}$ be the expected number of subtrees with j keys with root at level k (the root is at level 0). When an insertion falls in a subtree of size $j < 2t$, it is transformed to one of size $j+1$. If $j = 2t$ it is transformed with probability p_j into two subtrees of sizes $j-1$ and $2t+1-j$. The probability that an insertion in a tree of n elements falls in a subtree of size j at level k is $\frac{j+1}{n+1} A^j_{n,k}$. Hence, the following recurrence equations model the insertion process:

$$A^i_{n+1,k} = A^i_{n,k} + \frac{1}{n+1}(iA^{i-1}_{n,k} - (i+1)A^i_{n,k} + q_{i+1}(2t+1)A^{2t}_{n,k-1})$$

for $i = 0, \ldots, 2t$ and $q_i = p_i + p_{2t+2-i}$. Introducing generating functions on the index k writing in matrix form, and defining

$$H(z) = \begin{bmatrix} -1 & 0 & . & . & . & . & (2t+1)zq_1 \\ 1 & -2 & . & . & . & . & (2t+1)zq_2 \\ 0 & 2 & . & . & . & . & . \\ . & 0 & j & -(j+1) & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & (2t+1)zq_{2t} \\ . & . & . & . & 0 & 2t & -(2t+1)(1-zq_{2t+1}) \end{bmatrix}$$

we obtain

$$\vec{A}_{n+1}(z) = (I + \frac{1}{n+1}H(z))\vec{A}_n(z) ,$$

with the initial condition $\vec{A}_0(z) = (1, 0, ..., 0)$ and with I being the $(2t+1) \times (2t+1)$ identity matrix.

The average unsuccessful search time is given by [PM85, p.347] as

$$C'_n = \lambda'_1(1)H_{n+1} + O(1)$$

where $\lambda_1(z)$ is the eigenvalue of $H(z)$ that satisfies $\lambda_1(1) = 1$. Let $c_t = \lambda'_1(1)$. This derivative is calculated using the characteristic polynomial of $H(z)$ [14]:

$$p(\lambda, z) = \prod_{i=1}^{2t+1}(\lambda + i) - z(2t+1)! \sum_{k=1}^{2t+1} \frac{q_k}{(k-1)!} b_k(\lambda)$$

where

$$b_k(\lambda) = \begin{cases} 1 & k = 1 \\ \prod_{j=1}^{k-1}(\lambda + j) & k > 1 . \end{cases}$$

Computing c_t, we obtain

$$c_t = \frac{1}{H_{2t+2} - 1 - \frac{1}{2t+2}\left(\sum_{k=2}^{2t+1} kq_k(H_k - 1)\right)} .$$

Completing the sum $\sum_{k=2}^{2t+1} kq_k$ with the $k = 1$ term, we have

$$c_t = \frac{1}{H_{2t+2} - \frac{1}{2t+2}\sum_{k=1}^{2t+1} kq_k H_k} .$$

Considering some typical distributions for the p_i, we have
Uniform: $p_i = \frac{1}{2t+1}$ and $c_t = 2$ as in a normal binary search tree.
Binomial: $p_i = \binom{2t}{i-1}2^{-2t}$ and

$$c_t = \frac{1}{\ln 2 - \frac{2t}{(2t+1)(2t+2)} + \epsilon} ,$$

where $| \epsilon | < \frac{2^{-2t}}{t+1}$ [Sed75].

Using an analogy with Quicksort, the optimal distribution for the probabilities in a random tree is [Sed75, Theorem 8.1, p.221]: $p_{t+1} = 1$ and $p_j = 0$ for $j = 1, ..., 2t+1$ $(j \neq t+1)$. We have therefore proved the following:

Theorem 3.1. The optimal heuristic in a random tree is to choose the median. □

In this case, c_t takes its minimum value $c_t = 1/(H_{2t+2} - H_{t+1})$. The average successful search time can be obtained using the well known relation $C_n = (1 + 1/n)C'_n - 1$.

3.2 Height Balance Distribution

For the basic heuristic, equation (1) does not hold. However, we can show that equation (2) holds using the close relationship between Quicksort with a median of three elements and the basic heuristic. From

Sedgewick [Sed75], equation (1) for the basic heuristic is replaced by

$$\overline{hb}_k(n) = \frac{6}{n(n-1)(n-2)} \sum_{i=1}^{n} (n-i)(i-1)(\overline{hb}_k(i-1) + \overline{hb}_k(n-i)) \ .$$

Standard techniques for solving recurrences yield

$$\overline{hb}_k(n) = \frac{n+1}{n}\overline{hb}_k(n-1) + (n-7)\left(\frac{\overline{hb}_k(n-1)}{n} - \frac{\overline{hb}_k(n-2)}{n-1}\right) \ .$$

From here is easy to show by induction that

$$\overline{hb}_k(n) \geq \frac{n+1}{n}\overline{hb}_k(n-1)$$

and, hence, equation (2) is true. However, we can obtain a small improvement with the lower order terms of the exact solution for the previous recurrence. That is,

$$\frac{\overline{hb}_k(n)}{n+1} \geq \frac{\overline{hb}_k(j)}{j+1} + \left(\frac{\overline{hb}_k(j-8)}{j-7} - \frac{\overline{hb}_k(j-7)}{j-6}\right) \sum_{i=j+1}^{n+1} \frac{(j-7)^{\underline{8}}}{i^{\underline{8}}} \text{ for } n > j > 7 \qquad (3)$$

where $x^{\underline{n}} = x(x-1)\cdots(x-n+1)$ denotes a falling factorial, and the summation is bounded by a constant.

The values of $\overline{hb}_k$ for the case $t = 1$ are also shown in Table 1. In the basic heuristic the expected number of internal nodes with height balance 0 is bounded by

$$0.5734(n+1) \ \leq \ \overline{hb}_0(n) \ \leq \ 0.6503(n+1) \ .$$

The above improves the previous bounds of Bagchi and Reingold [BR82]. With the basic heuristic we have also that at least 85.6% of all internal nodes are roots with AVL height balance.

These results confirm the fact that random binary search trees are quite well balanced in the average, and even more so with the basic heuristic. A visual comparison of the height balance distribution is given by Figure 1.

3.3 Implementation Issues

The implementation for the basic heuristic ($t = 1$) that we propose is a modification of the implementation by Huang and Wong [HW83]. Huang and Wong store the two descendants of a node in the same record. Each record has two pointers, the left pointer addresses the two descendants of the left key and the right pointer addresses the two descendants of the right key. The nodes without siblings have empty space in their records. In the worst case the storage utilization for the basic heuristic is 75%. For the insertion and deletion algorithms we refer the reader to [HW83].

We can obtain a full utilization and one pointer per key in the worst case by storing the nodes without siblings (excluding the root) in a record with space for one key. To know the record type we need only one bit. In the worst case, all the keys are in records with pointers, i.e., one pointer per key. On the average 2/7 of the keys will not have siblings [HW83]. Hence, on the average we need space for n keys and $5n/7$ pointers. Therefore, if the key and the pointer have the same size, we can store a binary search tree saving more than

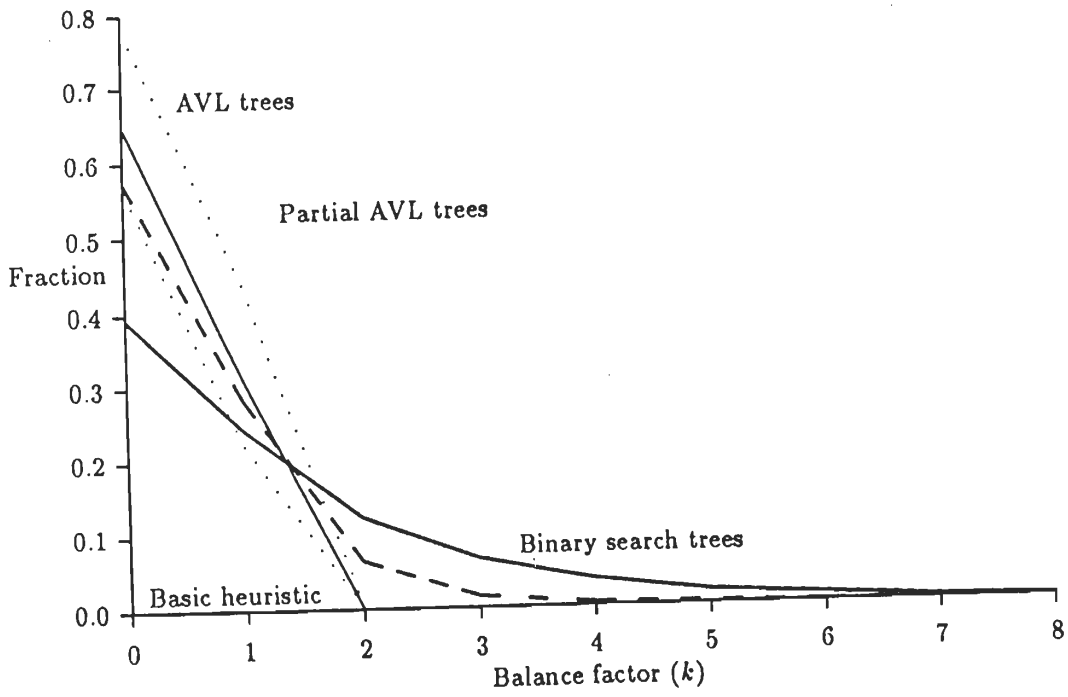

Figure 1: Height balance distribution.

42% of the space taken by the common implementation (n keys and $2n$ pointers). Also, the expected search time is reduced by 14%.

For $t > 1$, it is not clear which is the best way of keeping the subtrees with less than $2t + 1$ keys. Usually a binary search tree similar to the whole tree is used. This tree is reorganized when it reaches the critical size $2t + 1$. Another possibility is to use an ordered [Gre83] or unordered list. The choice depends on the value of t and the use of the tree (depending on the frequency of searches and insertions).

4 AVL Trees

AVL-trees, or height balanced trees, are binary search trees for which the height balance factor is at most one in each node [Knu73, GR91]. AVL-trees provide logarithmic worst cases for searches, insertions and deletions. The exact average case analysis for these trees is still open, although some bounds are known [Bro79, BYGZ90]. Here we extend the analysis by obtaining exact results for trees up to $n = 25$, extending the results of Richards [Ric83], and considering a variant of AVL trees that we call *j-partial AVL trees*. We define a j-partial AVL tree as a tree in which all subtrees of size less than j are AVL trees. For j-partial AVL trees we assume that equation (2) holds. Intuitively, this lower bound should be pessimistic.

For the generation of j-partial AVL trees only the symmetry at the root of the tree was used. The lower bound values for $\overline{hb}_k$ for j-partial AVL trees and bounds for AVL trees [BYGZ90] are given in Table 1. The exact curve for AVL trees lies within the dotted triangle.

For j-partial AVL trees with $j \geq 25$, the expected number of internal nodes with height balance 0 is bounded by

$$0.6483(n+1) \leq \overline{hb}_0(n) \leq 0.6884(n+1) .$$

Intuitively, these bounds should be also bounds for AVL trees. In this case the bounds would be better than Brown's [Bro79] and these of Baeza-Yates *et al.* [BYGZ90], namely

$$0.5637(n+1) \leq \overline{hb}_0(n) \leq 0.7799(n+1) .$$

Howeever, we are not able to prove that at least 64.83% of the nodes in an AVL tree are balanced nodes because rotations above subtrees of size j may change the height balance distribution.

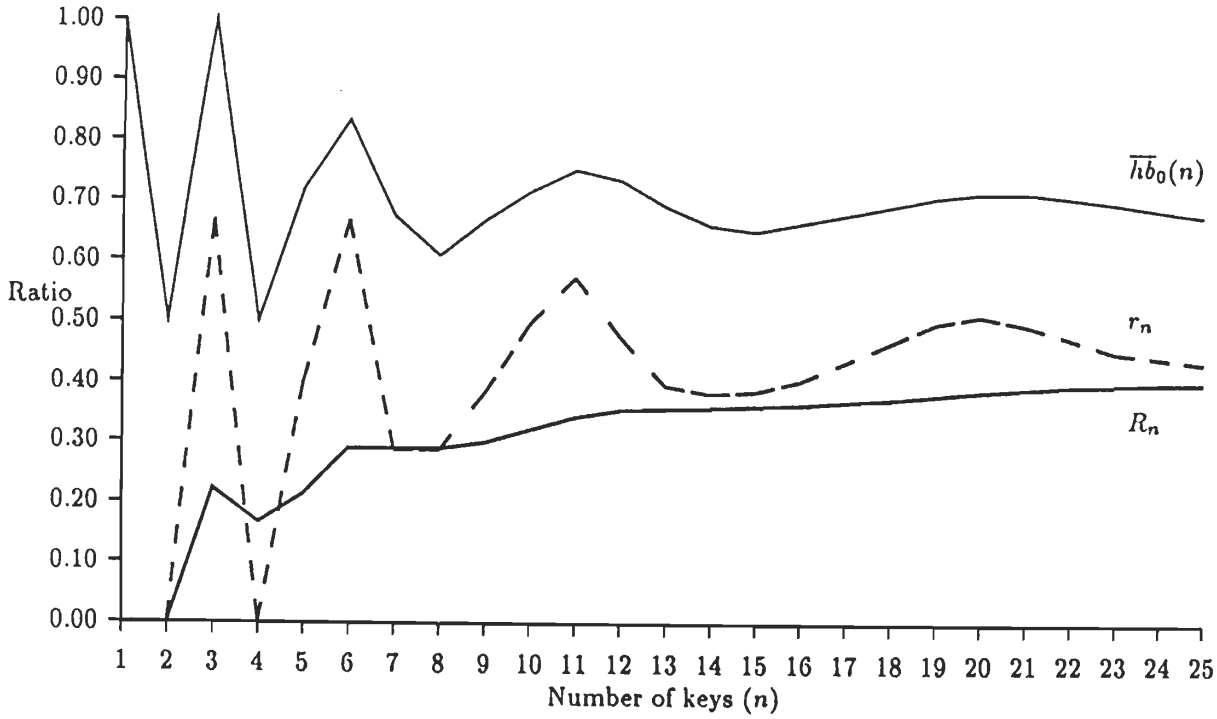

Figure 2: Fraction of balanced nodes and rotation probabilities.

To obtain more information about AVL trees, whose exact average case analysis is still an open problem, we generated the exact behavior all possible trees up to $n = 25$. For the generation of these trees only the symmetry at the root of the tree was used. This allows us to cut almost in half the number of trees considered by Richards [Ric83] who has results up to $n = 18$ (exactly in half for even n). Figures 2 and 3 (in a logarithmic scale and contrasted with totally balanced binary search trees) show these results for various parameters, where r_n is the probability of rotation in the n-th insertion, $R_n = \sum_{i=1}^{n} r_i/n$ is the average number of rotations per insertions, and H_n is the average height. These results suggest that the number of balanced nodes its approximately 69% (that is, closer to the upper bound of [BYGZ90]). Experimental results in [ZT82] give $r_n \approx 0.466$ and $C'_n \approx 1.018 \log_2 n + 0.051$.

Table 2 shows the new results for $n = 18$ to $n = 25$ that extends those by Richards [Ric83]. It also corrects the probability of no rotation for $n = 19$, and the fact that the average external path length (EPL)

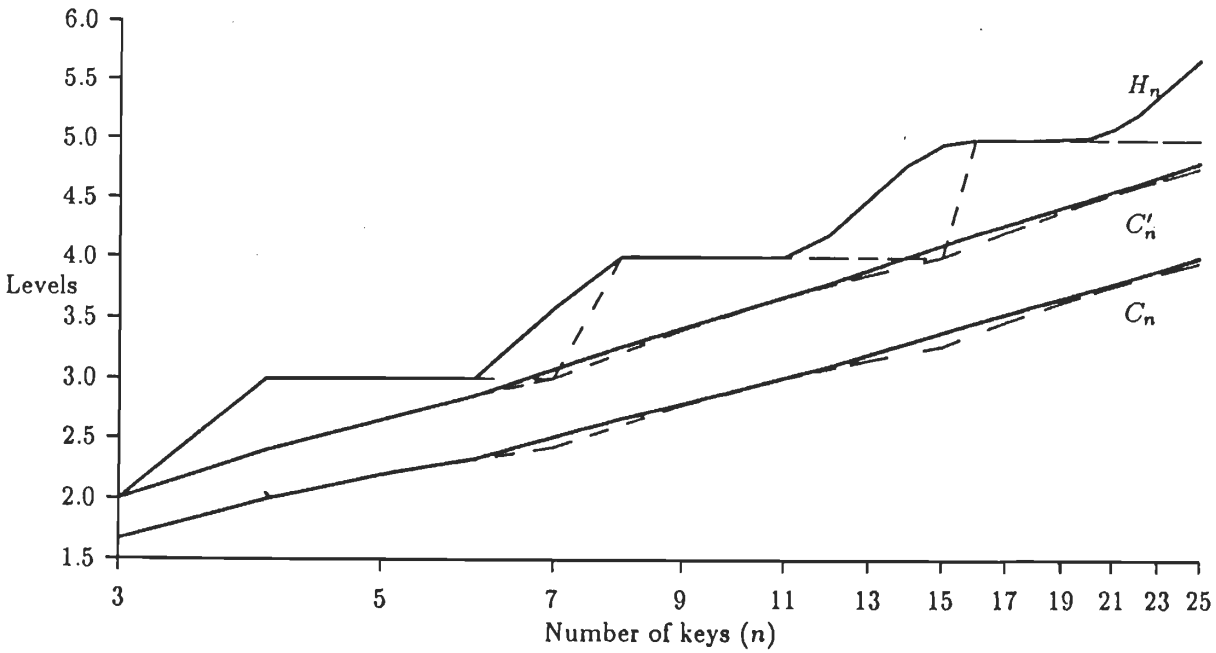

Figure 3: Average height and search costs for AVL trees (solid line) and totally balanced binary search trees.

is really the unsuccessful average search cost C'_n. We also include the average search cost for successful searches, the average height H_n, and the amortized number of rotations per insertion, R_n.

n	Tree shapes	Probabilities for the n-th insertion			R_n	C_n	C'_n	H_n
		No Rot	Single	Double				
18	3156	0.535819	0.232179	0.232003	0.371084	3.604358	4.362023	5.000000
19	5018	0.503246	0.248510	0.248244	0.377698	3.666485	4.433161	5.000000
20	8044	0.491279	0.254481	0.254240	0.384250	3.726225	4.501166	5.022732
21	19384	0.505706	0.247224	0.247074	0.389490	3.785124	4.567619	5.095702
22	41492	0.529744	0.235185	0.235071	0.393161	3.843387	4.632805	5.220293
23	88458	0.550126	0.224983	0.224890	0.395627	3.900551	4.696361	5.377563
24	173024	0.562511	0.218775	0.218714	0.397371	3.956237	4.757987	5.543058
25	335098	0.568649	0.215694	0.215657	0.398730	4.010248	4.817546	5.696355

Table 2: Exact results for small AVL trees.

Multiply by 2 to get A6265 with a different offset

Acknowledgments

The author wishes to acknowledge the helpful comments of Patricio Poblete, Gaston Gonnet, and Bob Dailey.

multiply by 2!
but not quite

References

[AVL62] G.M. Adel'son-Vel'skii and E.M. Landis. An algorithm for the organization of information. *Doklad Akademia Nauk SSSR*, 146(2):263–266, 1962.

[Bel65] C. Bell. *An Investigation into the Principles of the Classification and Analysis of Data on an Automatic Digital Computer.* PhD thesis, Leeds University, 1965.

[BR82] A. Bagchi and Edward M. Reingold. Aspects of insertion in random trees. *Computing*, 29:11–29, 1982.

[Bro79] M.R. Brown. A partial analysis of random height-balanced trees. *SIAM J on Computing*, 8(1):33–41, Feb 1979.

[BYGZ90] R. Baeza-Yates, G.H. Gonnet, and N. Ziviani. Expected behaviour analysis of AVL trees. *2nd Scandinavian Workshop in Algorithmic Theory, SWAT'90*, Springer-Verlag LNCS 447, pages 143–159, Bergen, Norway, July 1990.

[Gon83] Gaston H. Gonnet. Balancing binary trees by internal path reduction. *C.ACM*, 26(12):1074–1081, Dec 1983.

[GR91] G.H. Gonnet and Baeza-Yates R. *Handbook of Algorithms and Data Structures - In Pascal and C.* Addison-Wesley, Wokingham, UK, 1991. (second edition).

[Gre83] Daniel H. Greene. *Labelled Formal Languages and Their Uses.* PhD thesis, Computer Science Dept., Stanford University, June 1983.

[GS78] L.J. Guibas and Robert Sedgewick. A dichromatic framework for balanced trees. In *FOCS*, volume 19, pages 8–21, Ann Arbor MI, Oct 1978.

[HO85] L. Hermosilla and J. Olivos. A bijective approach to single rotation trees. In *5th International Conference in Computer Science*, pages 22–30, Santiago, Chile, 1985.

[HW83] S. Huang and C. Wong. Binary search trees with limited rotation. *BIT*, 23:436–455, 1983.

[IR80] A. Itai and M. Rodeh. Modified binary search trees. Technical Report 182, Dept. of Computer Science, Technion-IIT, Haifa, Israel, Aug 1980.

[Knu73] D.E. Knuth. *The Art of Computer Programming: Sorting and Searching*, volume 3. Addison-Wesley, Reading, Mass., 1973.

[NR73] Jurg Nievergelt and Edward M. Reingold. Binary search trees of bounded balance. *SIAM J on Computing*, 2(1):33–43, 1973.

[PM85] P.V. Poblete and J.I. Munro. The analysis of a fringe heuristic for binary search trees. *Journal of Algorithms*, 6:336–350, 1985.

[Ric83] R.C. Richards. Shape distribution of height-balanced trees. *Information Processing Letters*, 17:17–20, 1983.

[Sed75] R. Sedgewick. *Quicksort.* PhD thesis, Computer Science Department, Stanford University, May 1975. Report STAN-CS-75-492.

[WW76] W.A. Walker and Derick Wood. Locally balanced binary trees. *Computer Journal*, 19(4):322–325, Nov 1976.

[ZT82] N. Ziviani and F.W. Tompa. A look at symmetric binary B-trees. *Infor*, 20(2):65–81, May 1982.