Vol. 39 No. 4 Transactions of Information Processing Society of Japan Apr. 1998

Regular Paper

An Approximate Analysis of the AVL Balanced Tree Insertion Algorithm

RYOZO NAKAMURA,† NINGPING SUN†† and TAKUO NAKASHIMA†

Two particularly interesting questions concerning the performance of the AVL balanced tree insertion algorithm are: i) If all $n!$ permutations of n keys occur with the equal probability, what is the expected height of the constructed AVL balanced tree? ii) What is the probability that an insertion requires rebalancing? Although some empirical results about the expected height of the AVL balanced tree and the probability that an insertion requires rebalancing have been obtained, the mathematical analysis of these two problems is still open. In order to analyze both of these questions we should grasp the characters of the AVL balanced tree structures. In this paper, as the first step toward the analysis of these two open problems, we assume that all AVL balanced trees with the given number of nodes and height are constructed with equal probability instead of the hypothesis that all $n!$ permutations of n keys occur with equal probability. Based on this assumption, we first derive a formula to enumerate the different AVL balanced trees with n internal nodes and height h. Then, we propose the formulae to evaluate the expected height of the AVL balanced tree and the probability that an insertion requires rebalancing. The results from these proposed formulae are very close to those empirical ones obtained so far.

1. Introduction

The binary search tree technique is such a simple and efficient dynamic search method that it is qualified as one of the most fundamental and important algorithms. The search cost on the binary search trees is quite dependent on the shapes of the trees. In a binary search tree built from n random keys, the operations such as insertion and search require $O(\log\ n)$ units of time on the average, however, in the worst case, they take the linear time as $O(n)$. In order to eliminate the worst case performance of a binary search tree, several algorithms to balance the binary search tree were proposed and implemented [2),3)].

One of the most well known balanced tree algorithms is the AVL balanced tree which was proposed by two Russian mathematicians, Adel'son-Vel'skii and Landis [4)]. The balance criterion of the AVL balanced tree is: *A tree is called balanced if and only if for every node the heights of its two subtrees differ by at most 1.* In this paper, the height of a tree is defined to be the length of the longest path from the root to a leaf node.

According to the analysis of Adel'son-Vel'skii and Landis, the height of an AVL balanced tree with n internal nodes, denoted by $h(n)$, always lies between the heights of the perfectly balanced tree and the Fibonacci tree, i.e.,

$$\lceil \log_2(n+1) \rceil - 1 \leq h(n) \leq 1.4404 \log_2(n+1) - 1.328. \quad (1)$$

The lower bound of Eq. (1) denotes the height of the perfectly balanced tree, while the upper bound shows the height of the Fibonacci tree.

Some empirical results about the average behavior of the AVL balanced tree have been obtained, but no mathematical analyses of the following two questions about the AVL balanced tree insertion algorithm have been resolved successfully since 1962.

i) If all $n!$ permutations of n keys occur with the equal probability, what is the expected height of the constructed AVL balanced tree?

ii) What is the probability that an insertion requires rebalancing, i.e., how often do we need to do a single or a double rotation to restore the desired balance?

It is extremely difficult to mathematically analyze both of these problems in conformity with the hypothesis that all $n!$ permutations of n keys occur with equal probability. In order to solve these two questions it is indispensable to grasp the characters of the AVL balanced tree structures. In this paper, we take the first step toward analyzing the above two open problems by assuming that all AVL balanced trees

† Department of Computer Science, Faculty of Engineering, Kumamoto University
†† Graduate School of Science and Technology, Kumamoto University

with the given number of nodes and height are constructed with equal probability. Based on this assumption, we analyze both the expected height of AVL balanced tree and the probability that an insertion requires rebalancing. We first derive a formula to enumerate the different AVL balanced trees with n internal nodes and height h. Then, we propose the formulae to evaluate both of these problems. The results from the proposed formulae are very close to those empirical ones shown in Refs. 2) and 3).

2. Expected Height of the AVL Balanced Tree

In this section, according to our assumption that all AVL balanced trees with the given number of nodes and height are constructed with equal probability, we analyze the expected height of the AVL balanced tree with n internal nodes.

According to its definition, the AVL balanced tree with n nodes and height h can be constructed as **Fig. 1**. For a root node, if its left subtree is an AVL balanced tree with k nodes and height $h-1$ then its right subtree must be constructed as an AVL balanced tree with $(n-k-1)$ nodes and height $h-1$ or $h-2$. If we interchange the left and the right subtrees essentially the identical case will occur.

As we have mentioned, the optimum is reached if the tree is perfectly balanced for $n = 2^{h+1} - 1$. On the contrary, the worst AVL balanced tree is the Fibonacci tree, i.e., given the height of the tree, a Fibonacci tree is the AVL balanced tree with the minimum number of nodes. Therefore, the well-known definition of the Fibonacci tree is redefined as follows.

i) The height of the Fibonacci tree with only one node is 0.

ii) If the left subtree is the Fibonacci tree with height $h-1$ and the right subtree is the Fibonacci tree with height $h-2$, then the binary search tree is the Fibonacci tree with height h.

iii) Only when satisfied with the above two conditions, a binary search tree is a Fibonacci tree.

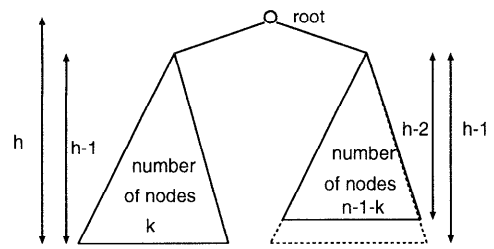


Fig. 1 The organization of the AVL balanced tree with n nodes and height h.

Let $f(h)$ be the number of nodes of the Fibonacci tree with height h, from the above definition, $f(h)$ can be expressed by the following recursive formula,

$$f(h) = f(h-1) + f(h-2) + 1, \tag{2}$$

where $f(0) = 1, f(1) = 2$.

The closed form expression of the recurrent formula (2) is given as follows[1],

$$f(h) = ((\Phi_1^{h+3} - \Phi_2^{h+3})/\sqrt{5}) - 1,$$

where $\Phi_1 = (1 + \sqrt{5})/2$, $\Phi_2 = (1 - \sqrt{5})/2$.

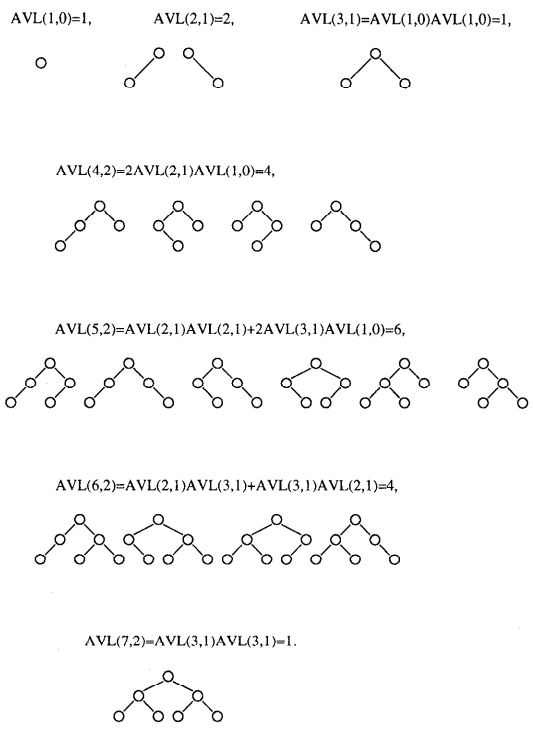
Now we consider how to count the number of the AVL balanced trees with n nodes and height h. Here, we inspect the range of k, the number of nodes in the left subtree of the root node. Since the Fibonacci tree is the AVL balanced tree with the minimum number of the nodes, the lower limit of k will be $f(h-1)$. On the other hand, for the upper limit of k, since it is clear that the left subtree with height $h-1$ must have no more than $2^h - 1$ nodes, and based on the balance criterion of the AVL balanced tree, the height of the right subtree of the root node is not less than $h-2$, i.e., the minimum number of nodes of the right subtree is $f(h-2)$. Therefore, the upper limit of k should be $\min\{2^h - 1, n - 1 - f(h-2)\}$. From the above analysis, the number of the AVL balanced trees with n nodes and height h, denoted by $AVL(n,h)$, can be expressed by the following recursive formula,

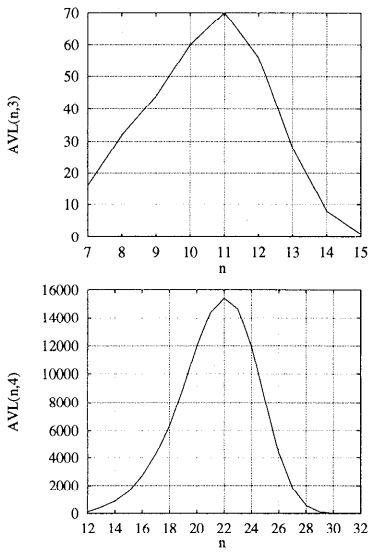
$$\begin{aligned} AVL(n,h) = & \sum_{k=f(h-1)}^{\min\{2^h-1,\, n-1-f(h-2)\}} AVL(k, h-1) \\ & \times \{AVL(n-k-1, h-1) \\ & + 2AVL(n-k-1, h-2)\}, \end{aligned} \tag{3}$$

where $n \geq 3, AVL(1,0) = 1, AVL(2,1) = 2$.

Based on Eq. (3), we can count the number of the AVL balanced trees constructed with n nodes and height h. Some results obtained by Eq. (3) are shown in **Fig. 2** in accordance with the different heights of the AVL balanced trees respectively, and tree structures of some simple AVL balanced trees are also presented there. From these figures, we can see that the number of the AVL balanced trees increases explosively with increasing of the number of nodes.

As a result of our evaluation, the expected





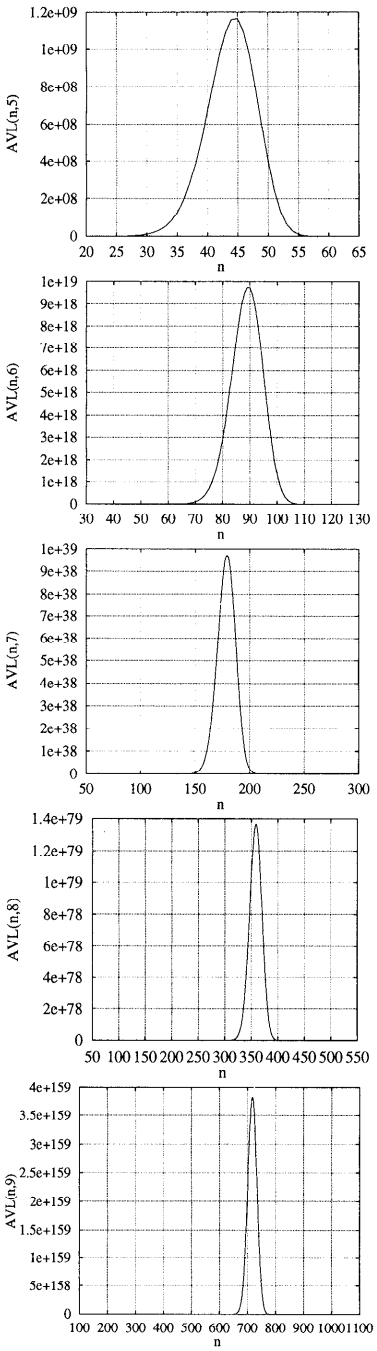


Fig. 2 The number of the AVL balanced trees with n nodes and height h.

height of the AVL balanced trees built from n nodes can be derived by $EAVL(n)$ as follows,

$$EAVL(n) = \sum_h hAVL(n,h) \Big/ \sum_h AVL(n,h). \quad (4)$$

where $\lceil \log_2(n+1) \rceil - 1 \leq h \leq 1.44 \log_2(n+1) - 1.328$.

Futhermore, we denote the average number of nodes in the AVL balanced tree with height h as $NAVL(h)$,

$$NAVL(h) = \sum_n nAVL(n,h) \Big/ \sum_n AVL(n,h). \quad (5)$$

We show the expected height and the average number of nodes of the AVL balanced trees ob-

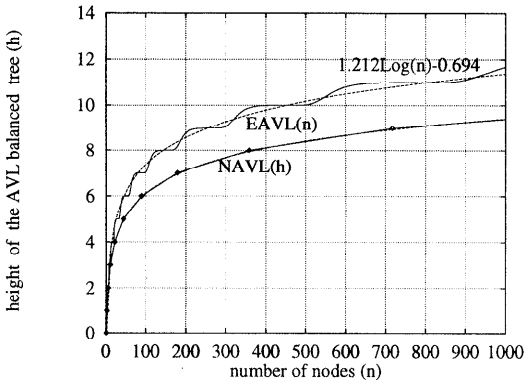


Fig. 3 The expected height and the average number of nodes of the AVL balanced tree.

tained by Eqs. (4) and (5) with the solid lines in **Fig. 3**. Using the method of least squares in the regression analysis [5], we can see quantitatively that the expected height of the AVL balanced tree approximates to $1.212 \log_2 n - 0.694$ and the average number of nodes approximates to $(1.459)2^h$ which are drawn by the dotted lines.

3. Probability that an Insertion Requires Rebalancing

In this section, we also assume that all AVL balanced trees with the given number of nodes and height are constructed with equal probability, and propose the analysis for the second problem: what is the probability that an insertion requires rebalancing?

When a new key is inserted into the AVL balanced tree using the tree insertion algorithm [2),3)], there are only two essentially different cases that will cause imbalance, which are represented in **Fig. 4** (a). Two other essentially identical cases will occur if we reflect the two diagrams of Fig. 4 (a) by interchanging the left and the right subtrees. In these diagrams the rectangles α, β, γ, δ represent the subtrees with the respective heights, and the heights added by the insertions are indicated by the crosses.

In case 1, using a single rotation we simply rotate the tree to the right, attaching β to B instead of A. While in case 2, we use a double rotation, first rotating (A, B) left then (B, C) right. These simple transformations restore the desired balance shown in Fig. 4 (b).

Because the structure of the AVL balanced tree is very complex, we can not easily find the probability that an insertion requires rebalancing only based on these two cases. Our careful scrutiny of the two cases in Fig. 4 reveals that there are only two essentially different patterns in the AVL balanced tree, called pattern-1 and pattern-2. As shown in **Fig. 5** the heights added by the insertions are indicated by the crosses.

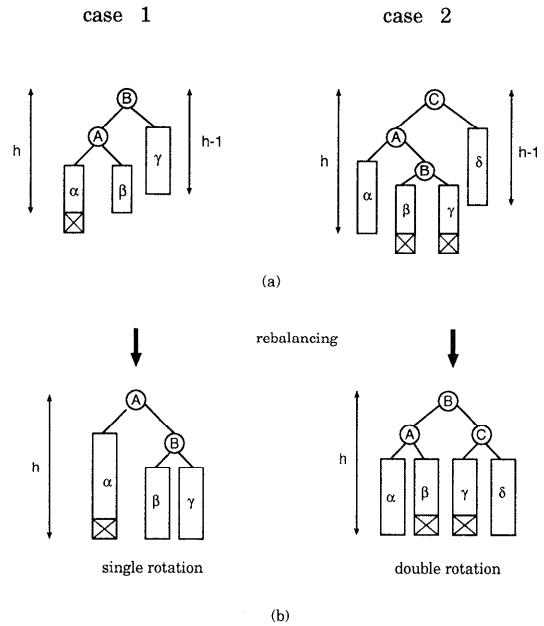


Fig. 4 Imbalance resulting from the insertion and restoring the balance.

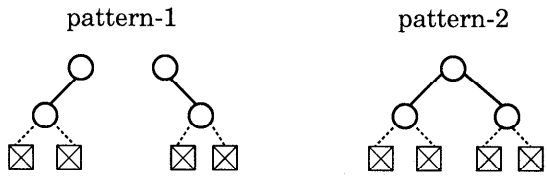


Fig. 5 Two essential patterns in an AVL balanced tree.

In other words, when we insert a new key into the external nodes of the AVL balanced tree with n nodes and height h, only these two patterns should be associated with the rebalancing shown in Fig. 4. According to these two patterns, there are only two different cases that the imbalance will happen due to an insertion as follows.

1. **imbalance I**
   The "imbalance I" indicates that an insertion at the external nodes of the leaf of pattern-1 causes the AVL balanced tree imbalance.
2. **imbalance II**
   The "imbalance II" indicates that an in-

sertion at the external nodes of pattern-2 causes the AVL balanced tree imbalance.

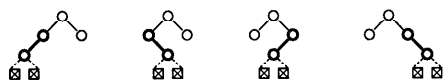
In order to find out the probabilities that these two imbalance cases occur, we introduce three functions into our analysis.

First, in the case of imbalance I, we denote $p_1(n,h)$ as the average number of pattern-1 in the AVL balanced trees with n nodes and height h by thc following recursive formula,

$$\begin{aligned} p_1(n,h) = \frac{1}{AVL(n,h)} \sum_{k=f(h-1)}^{\min\{2^h-1,\,n-1-f(h-2)\}} & AVL(k,h-1) \\ & \times[AVL(n-k-1,h-1) \\ & \times\{p_1(k,h-1)+p_1(n-k-1,h-1)\} \\ & +2AVL(n-k-1,h-2) \\ & \times\{p_1(k,h-1)+p_1(n-k-1,h-2)\}], \quad (6) \end{aligned}$$

where $n \geq 2$ and the initial value of $p_1(n,h)$ is $p_1(2,1)=1$.

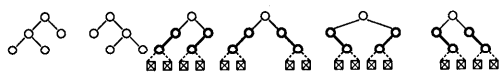
For instance, $p_1(4,2)$ is the average number of pattern-1 in the AVL balanced tree with 4 nodes and height 2, since AVL (4,2) is 4 shown as follows,



by Eq. (6) we get,

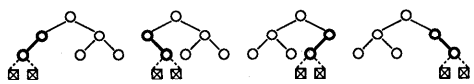
$$\begin{aligned} p_1(4,2) &= \frac{1}{AVL(4,2)} 2AVL(2,1)AVL(1,0) \\ &\quad \times\{p_1(2,1)+p_1(1,0)\} \\ &= \frac{1}{4}\cdot 2\cdot 2\cdot 1\{1+0\} = 1. \end{aligned}$$

Similarly, $p_1(5,2)$ is the average number of pattern-1 in the AVL balanced tree with 5 nodes and height 2, it becomes



$$\begin{aligned} p_1(5,2) &= \frac{1}{AVL(5,2)} AVL(2,1)AVL(2,1) \\ &\quad \times\{p_1(2,1)+p_1(2,1)\} \\ &= \frac{1}{6}\cdot 2\cdot 2\{1+1\} = \frac{8}{6} = \frac{4}{3}. \end{aligned}$$

And in the AVL balanced tree with 6 nodes and height 2,



the average number of pattern-1 is

$$\begin{aligned} p_1(6,2) &= \frac{1}{AVL(6,2)} 2AVL(2,1)AVL(3,1) \\ &\quad \times\{p_1(2,1)+p_1(3,1)\} \\ &= \frac{1}{4}\cdot 2\cdot 2\cdot 1\{1+0\} = 1. \end{aligned}$$

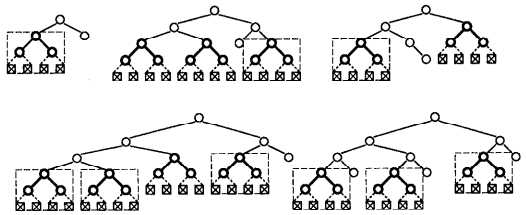


Fig. 6 The instances of imbalance II.

Secondly, we consider the cases of imbalance II. Not all pattern-2 in the AVL balanced trees are associated with imbalance II. Only those in the subtrees which are higher than their adjacent subtrees could possibly cause imbalance II. We show some tree structure instances of **Fig. 6** to explain this furthermore.

Obviously, imbalance II occurs only when a new key is inserted into the external nodes of those pattern-2 drawn within the dotted square. Therefore, in order to compute the probability that imbalance II occurs, we only concentrate on the average number of pattern-2 that will cause imbalance II.

According to the case of imbalance II, we intruduce the recursive functions $p_2(n,h)$ and $p_3(n,h)$ into our analysis now. By $p_2(n,h)$ we denote the average number of pattern-2 that will cause imbalance II in the AVL balanced tree with n nodes and height h.

$$\begin{aligned} p_2(n,h) = \frac{1}{AVL(n,h)} \sum_{k=f(h-1)}^{\min\{2^h-1,\,n-1-f(h-2)\}} & AVL(k,h-1) \\ & \times[AVL(n-k-1,h-1) \\ & \times\{p_2(k,h-1)+p_2(n-k-1,h-1)\} \\ & +2AVL(n-k-1,h-2) \\ & \times\{p_3(k,h-1)+p_2(n-k-1,h-2)\}], \quad (7) \end{aligned}$$

where $n \geq 5, h \geq 2$ and the initial value of $p_2(n,h)$ is $p_2(5,2) = 1/3$.

Here, by $p_3(n,h)$ we denote the average number of pattern-2 in the AVL balanced tree with n nodes and height h, where for every node, if the heights of its two subtrees are the same, all pattern-2 in both of the subtrees are counted; if the heights of its two subtrees are different, all pattern-2 in the higher subtree are counted, however for the lower subtree , only the pattern-2 that will cause imbalance II is counted. These two different cases are expressed in **Figs. 7** (a)

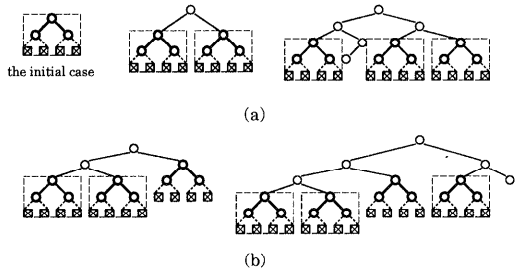


Fig. 7 The instances that pattern-2 is counted as $p_3(n,h)$ is called.

and (b) respectively, where those pattern-2 drawn within the dotted square are counted when $p_3(n,h)$ is called.

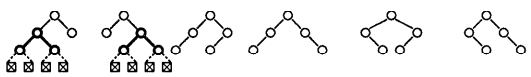
Therefore, function $p_3(n,h)$ is derived as follows,

$$\begin{aligned} p_3(n,h) = \frac{1}{AVL(n,h)} \sum_{k=f(h-1)}^{\min\{2^h-1,\,n-1-f(h-2)\}} & AVL(k,h-1) \\ & \times[AVL(n-k-1,h-1) \\ & \times\{p_3(k,h-1)+p_3(n-k-1,h-1)\} \\ & +2AVL(n-k-1,h-2) \\ & \times\{p_3(k,h-1)+p_2(n-k-1,h-2)\}], \quad (8) \end{aligned}$$

where $n \geq 3, h \geq 1$ and the initial value of $p_3(n,h)$ is $p_3(3,1)=1$.

Notice that, for every node, if the heights of its two subtrees are different, $p_2(n,h)$ contains a reference to $p_3(n,h)$ which contains a reference to $p_2(n,h)$. Hence, they are indirectly recursive.

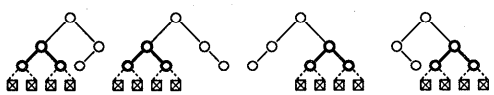
For example, in the AVL balanced tree with 5 nodes and height 2 shown as follows,



$$\begin{aligned} p_2(5,2) &= \frac{1}{AVL(5,2)} 2AVL(3,1)AVL(1,0) \\ &\quad \times\{p_3(3,1)+p_2(1,0)\} \\ &= \frac{1}{6}\cdot 2\cdot 1\cdot 1\{1+0\} = \frac{2}{6} = \frac{1}{3}, \\ p_3(5,2) &= \frac{1}{AVL(5,2)} 2AVL(3,1)AVL(1,0) \\ &\quad \times\{p_3(3,1)+p_2(1,0)\} \\ &= \frac{1}{6}\cdot 2\cdot 1\cdot 1\{1+0\} = \frac{2}{6} = \frac{1}{3}, \end{aligned}$$

here, $p_2(5,2)$ is equal to $p_3(5,2)$.

However, in the AVL balanced tree with 6 nodes and height 2,



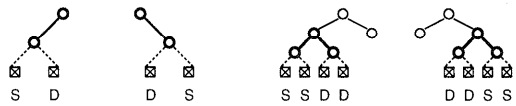
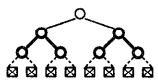


Fig. 8 Single and double rotations based on pattern-1 and pattern-2.

$$\begin{aligned} p_2(6,2) &= \frac{1}{AVL(6,2)} 2AVL(3,1)AVL(2,1) \\ &\quad \times\{p_2(3,1)+p_2(2,1)\} \\ &= \frac{1}{4}\cdot 2\cdot 1\cdot 2\{0+0\} = 0, \\ p_3(6,2) &= \frac{1}{AVL(6,2)} 2AVL(3,1)AVL(2,1) \\ &\quad \times\{p_3(3,1)+p_3(2,1)\} \\ &= \frac{1}{4}\cdot 2\cdot 1\cdot 2\{1+0\} = 1, \end{aligned}$$

even though there are 4 pattern-2's in these AVL balanced trees, the average number of pattern-2 that will cause imbalance II is zero.

Similarly, in the AVL balanced tree with 7 nodes and height 2,



$$\begin{aligned} p_2(7,2) &= \frac{1}{AVL(7,2)} AVL(3,1)AVL(3,1) \\ &\quad \times\{p_2(3,1)+p_2(3,1)\} \\ &= 1\cdot 1\{0+0\} = 0, \\ p_3(7,2) &= \frac{1}{AVL(7,2)} AVL(3,1)AVL(3,1) \\ &\quad \times\{p_3(3,1)+p_3(3,1)\} \\ &= 1\cdot 1\{1+1\} = 2, \end{aligned}$$

there are 2 pattern-2's in the AVL balanced tree with 7 nodes and height 2, but there are no any pattern-2 will cause imbalance II .

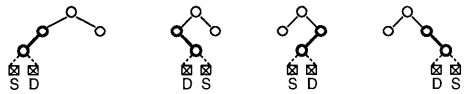
We are now in the position to analyze the probability that an insertion requires the rebalancing. As shown in Fig. 4, we use a single or a double rotation to restore the desired balance in both the imbalance cases. Based on the two proposed patterns, these rotations are indicated by two essentially different types of subtrees shown in **Fig. 8**, where S denotes a single rotation and D denotes a double rotation.

As shown in Fig. 8, the single rotation and the double rotation will happen with equal probability. Meanwhile, for a rotation, pattern-1 has one, while pattern-2 has two inserting positions. $PR(n,h)$ is denoted as the probability that a single or a double rotation should be used to restore the desired balance when a new key is inserted into the AVL balanced tree with n nodes and height h. It is derived according to

the imbalance cases mentioned previously,

$$PR(n,h) = \frac{1}{n+1}p_1(n,h) + \frac{2}{n+1}p_2(n,h). \quad (9)$$

For example, when a new key is inserted into the external node of the AVL balanced trees with 4 nodes and height 2,



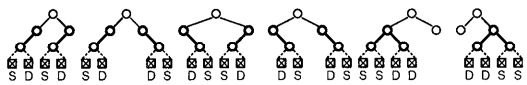
the probability $PR(4,2)$ becomes

$$PR(4,2) = \frac{1}{5}p_1(4,2) + \frac{2}{5}p_2(4,2)$$
$$= \frac{1}{5}\cdot 1 + \frac{2}{5}\cdot 0 = \frac{1}{5}.$$

Another example is

$$PR(5,2) = \frac{1}{6}p_1(5,2) + \frac{2}{6}p_2(5,2)$$
$$= \frac{1}{6}\cdot\frac{4}{3} + \frac{2}{6}\cdot\frac{1}{3} = \frac{1}{3},$$

its tree structures are:



In **Fig. 9** we show the probability that a single or a double rotation occurs when a new key is inserted into the AVL balanced tree with n nodes and height h. To present it more clearly, we show this probability with Figs. 9 (a) and (b) individually. From both these figures we can see that when the number of nodes is greater than 20 the probability that a single or a double rotation occurs is below 0.27.

Finally, let $P(n)$ be the probability that a single or a double rotation will occur when the $(n+1)$th key is inserted into the AVL balanced tree with n nodes. From the above analysis it can be defined as follows,

$$P(n) = \frac{1}{AVL(n)}\sum_h AVL(n,h)PR(n,h), \quad (10)$$

where $AVL(n) = \sum_h AVL(n,h)$, and

$\lceil \log_2(n+1) \rceil - 1 \le h \le 1.44\log_2(n+1) - 1.328.$

From Eq. (10), we can know when we insert $(n+1)$th key into the AVL balanced tree with n nodes how often we need to do a single or a double rotation. **Figure 10** gives the probabilities $P(n)$ that the $(n+1)$th key inserted requires a single or a double rotation to restore the balance. The results of Fig. 10 suggest that if the number of nodes is greater than 20 the value of $P(n)$ is between 0.26 and 0.27 on the average. Therefore, the necessary rebalancing should occur with about 0.54 probability, i.e., the rebalancing is required once for every two insertions. This result from $P(n)$ is very close to the ones from the empirical tests in Refs. 2) and 3).

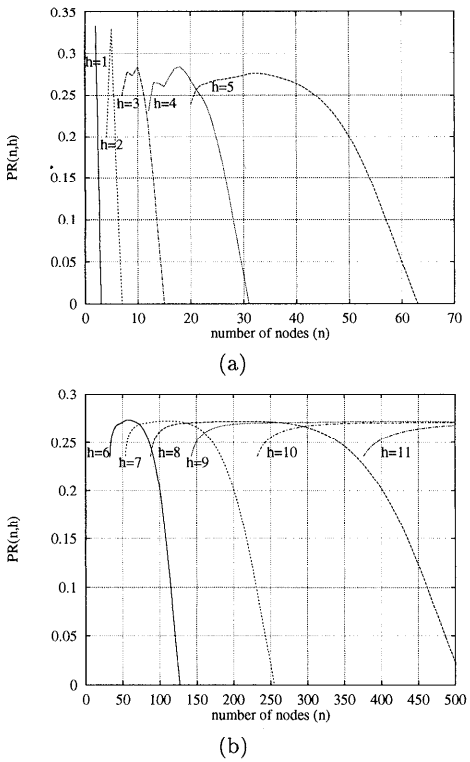


Fig. 9 The probability that a single or a double rotation occurs when a new key is inserted into the AVL balanced tree with n nodes and height h.

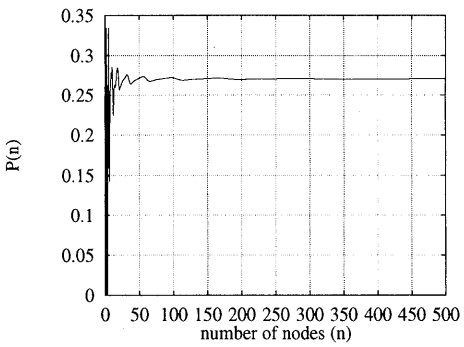


Fig. 10 The probability that a single or a double rotation occurs when a new key is inserted into the AVL balanced tree with n nodes.

4. Conclusions

Based on our assumption that all AVL balanced trees with the given number of nodes and height are constructed with equal probability we have analyzed two interesting questions concerning the average performance of the AVL balanced tree insertion algorithm: the expected height of the AVL balanced trees and the probability that an insertion requires the rebalancing. The proposed formulae can analyze approximately the performance of these two problems.

We have first derived the formula counting the number of the AVL balanced trees with n nodes and height h, i.e., $AVL(n,h)$, then got the expected height of the AVL balanced tree. Next, to evaluate the second problem we have revealed that the traditional cases causing imbalance correspond to the two imbalance cases, which are proposed based on the two essential patterns. The results computed by the proposed formulae are very close to the ones from the empirical tests.

It is well-known that all the AVL balanced trees should not be constructed with equal probability if all $n!$ permutations of n keys occur with equal probability. This is the reason why our results differ slightly from those empirical ones obtained so far. Therefore, based on our proposed analysis, we should and could exactly analyze both the expected height of the AVL balanced trees and the probability that an insertion requires the rebalancing in conformity with the assumption that all $n!$ permutations of n keys occur with the equal probability.

References

1) Knuth, D.E.: *The Art of Computer Programming*, Vol.1, Fundamental Algorithms, Addison-Wesley, Reading, MA (1973).
2) Knuth, D.E.: *The Art of Computer Programming*, Vol.3, Sorting and Searching, pp.451–469, Addison-Wesley, Reading, MA (1973).
3) Wirth, N.: *Algorithms + Data Structures = Programs*, pp.216–226, Prentice-Hall (1976).
4) Adel'son-Vel'skii, G.M., Landis, E.M.: *An Algorithm for the Organization of Information*, English translation in *Soviet Math.* Vol.3, pp.1259–1263 (1962).
5) Draper, N.R. and Smith, H.: *Applied Regression Analysis*, John Wiley & Sons (1966).

(Received May 27, 1997)
(Accepted February 2, 1998)

Ryozo Nakamura received the M.E. degree from Kumamoto University in 1968 and the D.E. degree in computer science from Kyushu University in 1985. From 1968 to 1974, he joined Chubu Electric Power Company. Since 1975 he has joined in Faculty of Engineering of Kumamoto University, and is presently a professor in Department of Computer Science. His current research interests include the design and analysis of algorithms and data structures.

Ningping Sun received her B.E. and M.E. from Beijing Polytechnic University and Kumamoto University. From 1983 to 1992 she joined Beijing Information Technology Institute as an instructor. She is presently working for her doctoral degree in computer science in the Graduate School of Science and Technology, Kumamoto University. She is interested in the design and analysis of algorithms and data structures, operating systems.

Takuo Nakashima received the M.E. degree from Kumamoto University in 1986. From 1986 to 1988 he joined Fujitsu Company. Since 1991 he has joined in the Faculty of Engineering, Kumamoto University, and is presently a research associate in Department of Computer Science. His research interests include the design and analysis of algorithms and data structures, and computer network.