Available online at www.sciencedirect.com
ScienceDirect
AKCE International Journal of Graphs and Combinatorics 15 (2018) 14–21
AKCE International Journal of Graphs and Combinatorics
www.elsevier.com/locate/akcej

Gap terminology and related combinatorial properties for AVL trees and Fibonacci-isomorphic trees☆

Mahdi Amani*

Department of Computer and Information Science, NTNU Norwegian University of Science and Technology, Trondheim, Norway
IT+Robotics srl, Prima Strada, 35, 35129 Padova PD, Italy
Department of Computer Science, University of Pisa, Pisa, Italy

Received 18 January 2017; received in revised form 22 January 2018; accepted 31 January 2018
Available online 12 February 2018

Abstract

We introduce *gaps* that are edges or external pointers in AVL trees such that the height difference between the subtrees rooted at their two endpoints is equal to 2. Using gaps we prove the *Basic-Theorem* that illustrates how the size of an AVL tree (and its subtrees) can be represented by a series of powers of 2 of the heights of the gaps, this theorem is the first such simple formula to characterize the number of nodes in an AVL tree. Then, we study the extreme case of AVL trees, the perfectly unbalanced AVL trees, by introducing *Fibonacci-isomorphic trees* that are isomorphic to *Fibonacci trees* of the same height and showing that they have the maximum number of gaps in AVL trees. Note that two *ordered trees* (such as AVL trees) are *isomorphic* iff there exists a one-to-one correspondence between their nodes that preserves not only adjacency relations in the trees, but also the roots. In the rest of the paper, we study combinatorial properties of Fibonacci-isomorphic trees.

Keywords: AVL tree; Fibonacci tree; Fibonacci-isomorphic tree; Gaps in AVL tree; Encoding

1. Introduction

AVL trees [2], presented in almost all books of algorithms and data structures as the perfect dictionary, are well investigated in different fields. An AVL tree ("Adelson-Velskii and Landis' tree", named after the inventors) is a *self-balancing* binary search tree, it was the first such data structure to be invented. In an AVL tree, the heights of the two child subtrees of any node differ by at most one; if at any time they differ by more than one, *rebalancing operation*

Peer review under responsibility of Kalasalingam University.
☆ A preliminary version of this paper appeared in the proceeding of International Conference on Current Trends in Graph Theory and Computation (CTGTC 2016) (Amani, 2017) [1].
* Correspondence to: Department of Computer and Information Science, NTNU Norwegian University of Science and Technology, Trondheim, Norway.
E-mail address: mahdi.amani@ntnu.no.
URL: http://pages.di.unipi.it/amani.

https://doi.org/10.1016/j.akcej.2018.01.019

is done to restore this property. The rebalancing operations of AVL trees are extremely elegant and powerful [3,4]. Since the invention of AVL trees in 1962, a wide variety of balanced binary search trees have been proposed, such as *red–black* [5] or *2–3 trees* [6] *weight-balanced trees* [7] just to cite the most popular ones, having remarkable properties, but none of them reaching the appeal of the original ones.

The first key contribution of this paper is the *Basic-Theorem* that illustrates how the size of an AVL tree (and its subtrees) can be represented by a series of powers of 2 of the heights of the *gaps*. *Gaps* are special edges or special external pointers in AVL trees such that the height difference between the subtrees rooted at the two endpoints of each gap is equal to 2. Basic-Theorem characterizes the size of any AVL tree with a very simple and interesting formula. Such a general formula, valid for the size of all AVL trees, was not known before. The Basic-Theorem and its corollaries seem powerful: for example, we can correlate the heights of the nodes with the heights of the gaps, or we can represent subtree sizes of a given AVL tree with summation of powers of 2 of the heights of the gaps, etc.

The second key contribution of this paper is studying the worst-case AVL trees that are the most unbalanced AVL trees and they have the maximum possible number of gaps with many appealing properties. To study these trees, we define the class of *Fibonacci-isomorphic trees* as the set of all ordered trees that are *isomorphic* to a *Fibonacci tree*, and we show that they represent the most unbalanced trees in the family of AVL trees. *Fibonacci trees* are AVL trees that in every branch, the height of the left subtree is greater than the height of the right one [8]. *Isomorphism on ordered trees* is a bijection between their node sets such that it preserves not only adjacency relations, but also the roots of the trees (for more detailed definitions, see Section 4). A preliminary version of this paper appeared in the proceeding of International Conference on Current Trends in Graph Theory and Computation (CTGTC 2016) (Amani, 2017) [1].

Paper organization. The paper is organized as follows. In Section 2 we define gap terminology in AVL trees. Basic-Theorem and its corollaries will be presented in Section 3, and in Section 4, we study combinatorial properties of Fibonacci-isomorphic trees.

2. Gap terminology

Let T be an AVL tree of size n and v be a given node, in this paper, we use the following notations.

- $|T|$ and $V(T)$ denote the size of tree T and its set of nodes, respectively.
- T_v denotes the subtree rooted in v and $key(v)$ denotes its key.
- $p(v)$ and $child(v)$ denote parent and child of v, respectively.
- v_l and v_r denote the left and the right children of v, respectively. Similarly, T_l and T_r denote the left and the right subtrees of T, respectively.
- $h(v)$ denotes the height of T_v that is the number of nodes on the longest simple downward path from the v to a leaf, the height of an empty tree is defined 0 [9].
- $lev(v)$ denotes the number of the nodes in the path from v to the root.

Definition 1 (*[9]*)**.** The balance factor of v (also denoted by $b(v)$) is the difference in the heights of its two subtrees $(h(v_r) - h(v_l))$. The balance factor of nodes of an AVL tree may take one of the values $-1, 0, +1$ (by definition). A node is *balanced* (or *unbalanced*) if its balance factor is 0 (or ± 1).

Fig. 1 illustrates the balance factors in a given AVL tree, in this figure, black nodes are balanced while the blue nodes are unbalanced.

Definition 2. For any pair of nodes v and w that $v = p(w)$, the edge between v and w is called *gap* iff the height difference between v and w is equal to 2. For a node v with only one child, the external pointer pointing to the *null child* is also considered a *gap* (since the height of v is 2 while the height of its null child is 0).

Fig. 2 illustrates gaps in a given AVL tree, which are shown by red lines, note that two of these gaps are external pointers pointing to "null children".

Definition 3. If node v is the parent of node w and there is a gap g between v and w, we say that v has a *gap child* g, and v and w are the *parent* and the *child* of g, denoted by $p(g)$ and $child(g)$, respectively. We also define the height of g, denoted by $h(g)$, equal to the height of its child (*i.e.*, $h(g) = h(child(g))$). We also use $GAP(T)$ to denote the set of all the gaps in an AVL tree T.

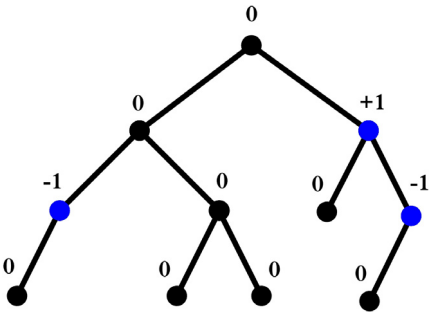

Fig. 1. Balance factor of nodes in a given AVL tree. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

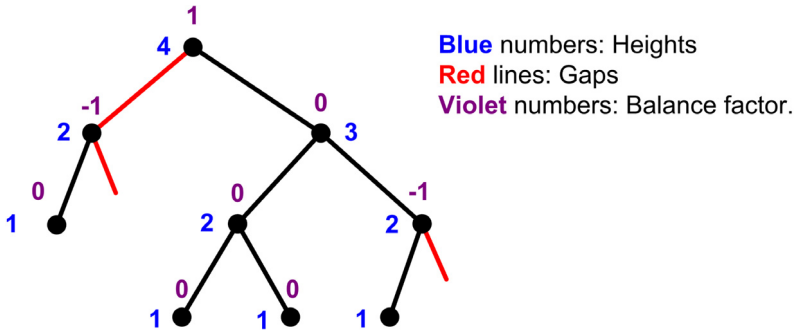

Fig. 2. An example of gaps in a given AVL tree. Gaps are shown by red lines. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Fact 1. *For any given node v in an AVL tree T:*

- *There is at most one gap child for v between v and v_l or v and v_r.*
- *Leaves have no gap children.*

3. General properties of gaps

In this section, we present Basic-Theorem that expresses the size of a given AVL tree T of height H in terms of the powers of 2 of the heights of the gaps.

Theorem 1 (*The Basic-Theorem*)**.**

$$|T| = n = 2^H - 1 - \sum_{g \in GAP(T)} 2^{h(g)}.$$

Proof. By induction on H. For $H = 0$ (empty tree) and $H = 1$ (single-node tree), the theorem trivially holds. For the inductive step on $H \geq 2$, we assume that the theorem holds for any AVL tree of height less than H, then we use this assumption to prove the statement for height H. We consider two cases for T.

First, suppose that the heights of T_l and T_r are equal to $H - 1$. Then, $GAP(T) = GAP(T_l) \cup GAP(T_r)$ as the two edges between the root of T and the roots of T_l and T_r are not gaps, and the theorem easily follows using the induction hypothesis,

$$\begin{aligned} |T| &= |T_l| + |T_r| + 1 \\ &= 2^{H-1} - 1 - \sum_{g \in GAP(T_l)} 2^{h(g)} + 2^{H-1} - 1 - \sum_{g \in GAP(T_r)} 2^{h(g)} + 1 \\ &= 2^H - 1 - \sum_{g \in GAP(T)} 2^{h(g)}. \end{aligned}$$

Now suppose that two subtrees have different heights $H-1$ and $H-2$, respectively. Then the set of the gaps of T contains all gaps in T_l and T_r, plus the new gap g' given by the edge between the root of T and the root of the subtree of height $H-2$ (*i.e.*, $GAP(T) = GAP(T_l) \cup GAP(T_r) \cup \{g'\}$). Therefore, using the induction hypothesis and the fact that $h(g') = H-2$, we have:

$$\begin{aligned}
|T| &= |T_l| + |T_r| + 1 = 2^{H-1} + 2^{H-2} - 1 - \sum_{g \in GAP(T_l)} 2^{h(g)} - \sum_{g \in GAP(T_r)} 2^{h(g)} \\
&= 2^H - 2^{H-2} - 1 - \sum_{g \in GAP(T_l)} 2^{h(g)} - \sum_{g \in GAP(T_r)} 2^{h(g)} \\
&= 2^H - 1 - 2^{h(g')} - \sum_{g \in GAP(T_l)} 2^{h(g)} - \sum_{g \in GAP(T_r)} 2^{h(g)} \\
&= 2^H - 1 - \sum_{g \in GAP(T)} 2^{h(g)}.
\end{aligned}$$

Thereby, the proof is complete. □

To show how powerful this theorem is, the following corollary describes the precise relation between the size of the entire tree (n), the heights of the nodes, the subtree sizes, and the heights of the gaps in a given AVL tree.

Corollary 1. *For a gap g, let $lev(g)$ be the number of the nodes above g in the path from g to the root (i.e., the number of node-ancestors of g), note that the level of a gap is equal to the level of its parent node, then:*

$$\sum_{u \in V(T)} (2^{h(u)} - |T_u|) - \sum_{g \in GAP(T)} lev(g) 2^{h(g)} = \boldsymbol{n}.$$

Proof. By Theorem 1 (Basic-Theorem) we know that for any node u, $2^{h(u)} - \sum_{g \in GAP(T_u)} 2^{h(g)} - |T_u| = 1$. Therefore, by summing up of this formula over all nodes we have:

$$\sum_{u \in V(T)} 1 = n = \sum_{u \in V(T)} \{2^{h(u)} - |T_u| - \sum_{g \in GAP(T_u)} 2^{h(g)}\}.$$

On the other hand, for any gap g, for any ancestor u of g, $g \in GAP(T_u)$ and vice versa. Therefore, there are exactly $lev(g)$ nodes u that $g \in GAP(T_u)$. We claim that:

$$\sum_{u \in V(T)} \sum_{g \in GAP(T_u)} 2^{h(g)} = \sum_{g \in GAP(T)} lev(g) 2^{h(g)}.$$

Therefore,

$$n = \sum_{u \in V(T)} (2^{h(u)} - |T_u|) - \sum_{u \in V(T)} \sum_{g \in GAP(T_u)} 2^{h(g)},$$

$$n = \sum_{u \in V(T)} (2^{h(u)} - |T_u|) - \sum_{g \in GAP(T)} lev(g) 2^{h(g)}. \quad \square$$

Corollary 2. *The powers of 2 of the heights of the nodes and the gaps are related by the following upper bound.*

$$\sum_{u \in V(T)} (2^{h(u)}) - \sum_{g \in GAP(T)} lev(g) 2^{h(g)} \leq n + nH = \Theta(n \log n).$$

Proof. Immediately by using Corollary 1 and the fact that $\sum_{u \in V(T)} (|T_u|)$ is the same as the total internal path length that is upper bounded by $nH \leq \Theta(n \log n)$. □

As we observed, Basic-Theorem is a very simple but interesting and unique formula that can be used in many different combinatorial problems. The above corollaries were just some primary examples, more can be found based on corresponding applications.

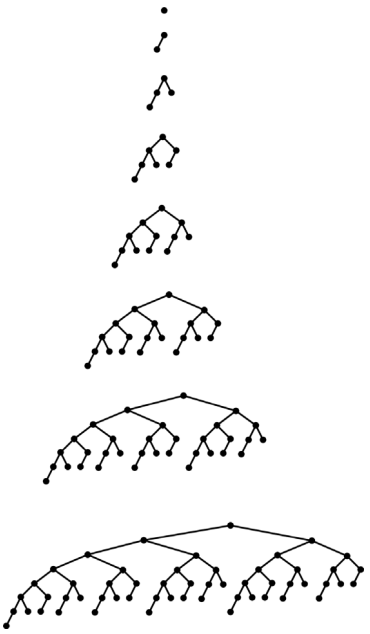

Fig. 3. The list of all Fibonacci trees of height 1 to 8.

4. Fibonacci-isomorphic trees

In this section, we first review the definition of *Fibonacci trees*, then, we define a new class of trees called *Fibonacci-isomorphic trees* showing that they represent the extreme case of perfectly unbalanced AVL trees and they have the maximum possible number of gaps, and finally, we study their combinatorial properties.

Fibonacci trees. *Fibonacci trees* are AVL trees that in every branch, the height of the left subtree is bigger than the height of the right one.[1] A *Fibonacci tree of height h* has F_h leaves, where F_i shows the ith Fibonacci number (*i.e.*, $F_0 = 0$, $F_1 = 1$, $F_i = F_{i-1} + F_{i-2}$). Fibonacci tree can be also defined recursively as follows [8].

Definition 4. The Fibonacci tree of height h for $h = 0$ is an empty tree and for $h = 1$ is just a single node; If $h \geq 2$, the left subtree of the Fibonacci tree of height h is the Fibonacci tree of height $h - 1$ and the right subtree is the Fibonacci tree of height $h - 2$.

We emphasis that for any given height h, there is a **unique** Fibonacci tree of height h. Fig. 3 illustrates Fibonacci trees of height 1 to 8.

Isomorphism on ordered trees. Recall that two "graphs" are isomorphic if there exists a one-to-one correspondence between their node sets that preserves adjacency relations in the graphs. For "ordered trees", *isomorphism* also preserves the roots (*i.e.*, roots are mapped to each other) [11]. More precisely, if T and T' are two ordered trees and V, E, r, V', E', r' denote the set of the nodes, the set of the edges, and the roots of T and T', respectively, *isomorphism of ordered trees* T and T' is a bijection between their nodes $f : V \to V'$ such that:

$$f(r) = r' \text{ and } \forall u, v \in V : (u, v) \in E \Leftrightarrow (f(u), f(v)) \in E'.$$

In simple words, two ordered trees are isomorphic if one tree can be obtained from the other by performing any number of *flips* while flip means swapping left and right children of a node. Fig. 4 shows two isomorphic ordered trees.

[1] We are familiar with the beauty of Fibonacci numbers (see [10]), in Fibonacci trees, by construction, a semi-Fibonacci pattern repeats in every branch.

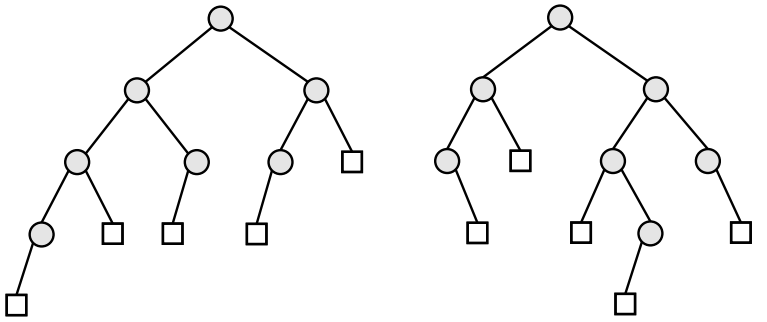

Fig. 4. A Fibonacci tree of height 5 in the left side and one of its isomorphisms in the right side.

Fibonacci-isomorphic trees. We define a *Fibonacci-isomorphic* tree as an ordered tree that is isomorphic to a Fibonacci tree. Fig. 4 also shows two Fibonacci-isomorphic trees of height 5, the left one is a Fibonacci tree of height 5 and the right one is one of its isomorphisms.

Corollary 3. *A Fibonacci-isomorphic tree is an AVL tree that all internal nodes have exactly one "gap child" and vice versa.*

Fact 2. *In a Fibonacci-isomorphic tree of height h,*

1. *its "internal nodes" form a Fibonacci-isomorphic tree of height* $h-1$,
2. *the total number of its nodes is* $|T| = \sum_{i=1}^{n} F_i = F_{h+2} - 1$.

Proof. The first statement can be proved by a simple induction on h. The second statement can be proved by another induction using the first statement. □

Lemma 1. *The number of Fibonacci-isomorphic trees of height h is* $2^{F_{h+1}-1}$.

Proof. For any internal node, by a flip we generate a new Fibonacci-isomorphic tree, on the other hand, by Fact 2, we have $F_{h+1} - 1$ "internal" nodes, therefore, the total number of Fibonacci-isomorphic trees of height h is $2^{F_{h+1}-1}$. □

4.1. Encoding and labeling

Based on what we mentioned earlier, the cardinality of the set of Fibonacci-isomorphic trees of height h is $2^{F_{h+1}-1}$. In the following, we present an optimum encoding for these trees using only $F_{h+1} - 1$ **bits**.

Encoding. Ignore all the leaves, for any internal node, label the node with its balance factor, then replace all -1 labels with 0 and all $+1$ labels with 1, finally, by a preorder traversal of T, obtain the codeword.

Algorithm 1 *Encoding Algorithm for Fibonacci-Isomorphic Trees:*
1: Label each internal node with its balance factor.
2: Replace every label $+1/-1$ with $1/0$, respectively.
3: Obtain the codeword by a preorder traversal.

Fig. 5 demonstrates encoding of a Fibonacci tree of height 5, in this picture, red lines represent gaps, black numbers show the balance factors, and the codeword is 0 1 0 0 0 1 0.

The validity of this encoding is trivial (the *validity* of an encoding is the one-to-one correspondence between the encoding and the tree) and since the encoding is optimum, it can be efficiently used for many combinatorial applications such as generation, ranking, unranking, random generation, etc. For instance, in the following, we demonstrate a generation algorithm of Fibonacci-isomorphic trees using this encoding.

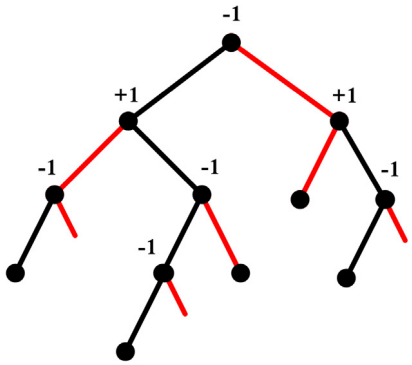

Fig. 5. Labeling a Fibonacci-isomorphic tree, in this picture, red lines show the gaps, black numbers show the balance factors, and the codeword is 0 1 0 0 0 1 0. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4.2. Generation of Fibonacci-Isomorphic trees

Many papers have been published earlier in the literature for generating different classes of trees. Just to name a few, we can mention the generation of binary trees in [12–14], k-ary trees in [15–17], degree bounded ordered trees in [18,19], and neuronal trees in [20–22]. Typically, trees are encoded as codewords over a given alphabet and then these codewords are generated, *e.g.* in our Fibonacci-isomorphic trees the presented codeword uses an alphabet of size 2.

By choosing a suitable codeword to represent the trees, we can design efficient generation algorithms for the codeword. Any generation algorithm imposes an ordering on the set of trees. Here, we define an ordering on these trees with a given height h that is relevant to the numerical generation of these codewords from $\underbrace{0, 0, \ldots, 0}_{F_{h+1}-1 \text{ copies}}$ (which is the *first* codeword in lexicographical order) to $\underbrace{1, 1, \ldots, 1}_{F_{h+1}-1 \text{ copies}}$ (which is the *last* codeword in lexicographical order).

Definition 5. Let T and T' be two Fibonacci-isomorphic trees, $h(T)$ shows the height of tree T, and T_l (T_r) and T'_l (T'_r) denote the left (right) subtrees of T and T', respectively. If $T = T'$, they have the same order, otherwise, we say that T is less than T' in this ordering and we show it by $T \prec T'$, iff

- $h(T) < h(T')$ or
- $h(T) = h(T')$ and recursively $T_l \prec T'_l$ or ($T_l = T'_l$ and $T_r \prec T'_r$).

Theorem 2. *Numerical generation of all* $2^{F_{h+1}-1}$ *codewords from* $\underbrace{0, 0, \ldots, 0}_{F_{h+1}-1 \text{ copies}}$ *to* $\underbrace{1, 1, \ldots, 1}_{F_{h+1}-1 \text{ copies}}$ *is equivalent to the generation of their corresponding Fibonacci-isomorphic trees in the ordering given in Definition 5.*

Proof. Immediate by construction. Note that, if after a sequence of the same elements in both codewords, there is a 0 in one codeword and a 1 in another codeword, it is equivalent to reaching to a subtree with its height less than the others (as their parents had the same heights). □

5. Summary and remarks

In this paper, we introduced *gaps* in AVL trees. Using gaps, we proved the *Basic-Theorem* that illustrates how the size of an AVL tree (and its subtrees) can be represented by a series of powers of 2 of the heights of the gaps. This theorem is the first such simple formula to characterize the number of nodes in an AVL tree. Then, we introduced *Fibonacci-isomorphic trees* showing that they are the extreme case of perfectly unbalanced AVL trees with maximum possible number of gaps, finally, we studied the basic combinatorial properties of Fibonacci-isomorphic trees such as enumeration, encoding, and generation.

Acknowledgments

I would like to express my very great appreciation to Prof. Anna Bernasconi, Prof. Roberto Grossi, and Prof. Linda Pagli for their encouragement, their contribution to this work, and their time and comments that greatly improved the manuscript.

References

[1] M. Amani, New terminology and results for avl trees, in: International Conference on Current Trends in Graph Theory and Computation, Electron. Notes Discrete Math. 63 (2017) 101–108. http://dx.doi.org/10.1016/j.endm.2017.11.004.
[2] G.M. Adel'son-Vel-skii, E. Landis, An algorithm for the organization of information, Sov. Math. Doklady 3 (1962) 1259–1263.
[3] K. Mehlhorn, A. Tsakalidis, An amortized analysis of insertions into AVL-trees, SIAM J. Comput. 15 (1) (1986) 22–33.
[4] M. Amani, K.A. Lai, R.E. Tarjan, Amortized rotation cost in avl trees, Inform. Process. Lett. 116 (5) (2016) 327–330. http://dx.doi.org/10.1016/j.ipl.2015.12.009.
[5] R. Bayer, Symmetric binary B-trees: Data structure and maintenance algorithms, Acta Inform. 1 (4) (1972) 290–306.
[6] L.J. Guibas, R. Sedgewick, A dichromatic framework for balanced trees, in: 19th Annual Symposium on Foundations of Computer Science, IEEE, 1978, pp. 8–21.
[7] J. Nievergelt, E.M. Reingold, Binary search trees of bounded balance, SIAM J. Comput. 2 (1) (1973) 33–43.
[8] Y. Horibe, Notes on Fibonacci trees and their optimality, Fibonacci Quart. 21 (2) (1983) 118–128.
[9] D.E. Knuth, The Art of Computer Programming: Sorting and Searching, Vol. 3, Pearson Education, 1998. http://dx.doi.org/10.1093/comjnl/17.4.324.
[10] P.S. Stevens, Patterns in Nature, Little Brown & Company, 1974.
[11] A. Jovanović, D. Danilović, A new algorithm for solving the tree isomorphism problem, Comput. J. 32 (1984) 187–198. http://dx.doi.org/10.1007/BF02243572.
[12] H. Ahrabian, A. Nowzari-Dalini, On the generation of binary trees in A-order, Int. J. Comput. Math. 71 (3) (1999) 351–357. http://dx.doi.org/10.1080/00207169908804813.
[13] H. Ahrabian, A. Nowzari-Dalini, Parallel generation of binary trees in a-order, Parallel Comput. 31 (8) (2005) 948–955. http://dx.doi.org/10.1016/j.parco.2005.06.002.
[14] R.-Y. Wu, J.-M. Chang, Y.-L. Wang, A linear time algorithm for binary tree sequences transformation using left-arm and right-arm rotations, Theoret. Comput. Sci. 355 (3) (2006) 303–314. http://dx.doi.org/10.1016/j.tcs.2006.01.022.
[15] K. Manes, A. Sapounakis, I. Tasoulas, P. Tsikouras, Recursive generation of k-ary trees, J. Integer Seq. 12 (2) (2009) 1–18. article 09.7.7.
[16] A. Ahmadi-Adl, A. Nowzari-Dalini, H. Ahrabian, Ranking and unranking algorithms for loopless generation of t-ary trees, Logic J. IGPL 19 (1) (2011) 33–43. http://dx.doi.org/10.1093/jigpal/jzp097.
[17] R.-Y. Wu, J.-M. Chang, C.-H. Chang, Ranking and unranking of non-regular trees with a prescribed branching sequence, Math. Comput. Modelling 53 (5) (2011) 1331–1335. http://dx.doi.org/10.1016/j.mcm.2010.12.019.
[18] M. Amani, A. Nowzari-Dalini, Generation, ranking and unranking of ordered trees with degree bounds, in: Proceedings of Eleventh International Workshop on Developments in Computational Models, Cali, Colombia, October 28, 2015, Electronic Proceedings in Theoretical Computer Science, 2015, pp. 31–45. http://dx.doi.org/10.4204/eptcs.204.4.
[19] M. Amani, A. Nowzari-Dalini, A new encoding for efficient generation, ranking and unranking of semi-chemical trees, in: DCM 2015, 2015, pp. 4–15.
[20] V. Vajnovszki, Listing and random generation of neuronal trees coded by six letters, Autom. Comput. Appl. Math. 4 (1) (1995) 29–40.
[21] M. Amani, A. Nowzari-Dalini, Ranking and unranking algorithm for neuronal trees in b-order, J. Phys. Sci. 20 (2015) 19–34.
[22] M. Amani, A. Nowzari-Dalini, H. Ahrabian, Generation of neuronal trees by a new three letters encoding, Comput. Inform. 33 (6) (2015) 1428–1450.