The Single Direction Edges in the Binary Tree

Adrijan Bozinovski
School of Computer Science and Information Technology at University American College Skopje
1000 Skopje, Macedonia
{bozinovski@uacs.edu.mk}

ABSTRACT: *Using the complete binary tree, we have introduced a generalized data structure and named it as Leveled binary tree. We do maintain the single direction during inserting nodes in the levelled tree, leads to the single direction edges in the binary tree. We observed that regularities are formalized using mathematical formulae, which are presented and proved, and it is shown how they produce specific integer sequences which can be expanded to infinity. The findings seem to generate further interest in this direction.*

Keywords: Leveled Binary Tree, Single-Direction Edges, Minimum, Maximum, Integer Sequence, Online Encyclopedia of Integer Sequences

Received: 30 August 2020, Revised 29 November 2020, Accepted 9 December 2020

DOI: 10.6025/jitr/2021/12/1/8-12

1. Introduction

Trees are fundamental concepts in computer science, and are frequently used to keep track of ancestors or descendants, sports tournaments, organizational charts of large corporations and so on [1]. Trees are one of the basic data structures used in combinatorial algorithms [2], search techniques (e.g. [3, 4]), and game playing [5]. This paper also points out the use of binary trees for generating integer sequences, which are important in information forensics [6], cryptography [7], and security [8].

A binary tree is a data structure made up of nodes, in which each node contains an information part and links, also called edges, to two other such nodes, called the node's left and right child nodes, respectively. A node can be null as well, in which case it contains no information. A recursive definition of a binary tree is that it is a structure with a finite set of nodes, which either contains no nodes or contains a root node and binary trees as its left and right child nodes [9].

Binary trees have been shown to be very useful in mathematics and computer science and as such have been extensively studied. Several variations of the binary tree structure have been conceived, such as binary search trees, red-black trees [9], AVL trees [10], B-trees [11], and so on. Binary trees are often used as auxiliary data structures in other research endeavors, both practical (e.g., [12,13]) and theoretical (e.g., [14,15]), but occasionally are the subject of the research itself (e.g., [16]).

In this paper, a new variation of binary trees, called leveled binary trees, will be introduced. It will be shown how particular integer sequences can be generated using them, which will be presented and the formulae for their generation will be proved. These integer sequences have been included into the Online Encyclopedia of Integer Sequences.

2. Leveled Binary Trees

A leveled binary tree is a binary tree in which nodes are inserted in a breadth-first fashion. In other words, insertion of a node increases the height of the tree only when the tree is full, i.e., when all of the positions at the last level have already been occupied. Figure 1 shows examples of leveled binary trees with numbers of *nodes n* = [1, 9]

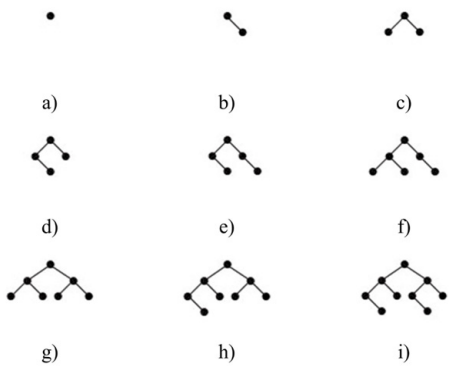

Figure 1. Examples of leveled binary trees with numbers of nodes n = [1,9]

In a leveled binary tree, nodes can be inserted arbitrarily, as long as the leveled structure is preserved, in a sense that no new level is inserted until all the possible positions in the last level have been occupied. This is a generalization of the complete binary tree structure, where nodes in the last level are always placed as far left as possible [17]. Also, if a leveled binary tree has $n = 2k - 1$ nodes, where $k \geq 0$ is an integer, it is a full binary tree [18]. Examples of full binary trees are shown in Figure 1a, 1c and 1g, having 1, 3 and 7 nodes respectively. In this paper, the root is treated as being placed on level 0.

3. Minimum Number of Single-direction Edges in a Leveled Binary Tree

3.1. Motivation and Formula

During the course of research, a question appeared about the regularity by which the minimum number of left edges appear in a leveled binary tree with progressively increasing values of n. Specifically, a question arose about how to obtain a formula which would produce the minimum numbers of left edges in leveled binary trees with n nodes.

Initially, this problem was solved algorithmically, and the formula was extracted subsequently. The formula is presented in equation (1).

$$a_{min}(n) = 2^{h-1} - 1 + (n - 3 \cdot 2^{h-1} + 1) \cdot H(n - 3 \cdot 2^{h-1} + 1) \qquad (1)$$

where $h = \lfloor \log 2n \rceil$ and represents the height of the leveled binary tree, and $H()$ is the Heaviside step function.

The following section provides a proof of this formula. For the sake of simplicity, the proof will concern the minimum number of left edges in a leveled binary tree with a given number of nodes n. The same formula may be used for the minimum number of right edges in a leveled binary tree, whereas the proof is analogous.

3.2. Proof of the Formula

Building a leveled binary tree is done in such a way that nodes are inserted only on a single level, until it is completely filled, i.e., inserting nodes in a new level is possible only when all levels up to the new one have been filled completely. Thus, the nature of the leveled binary tree guarantees that its level is always $h = \lfloor \log 2n \rceil$.

If a leveled binary tree has $n = 2k - 1$ nodes, it is a full tree. Since the number of edges in any tree is 1 less than the number of its nodes, a full binary tree contains $2k - 2$ edges. Since there is an equal number of left and right edges in a full binary tree, 2k-1 - 1 edges in a full binary tree are strictly left (and also strictly right).

If a leveled binary tree has $n \neq 2k - 1$ nodes, it is not a full tree. However, since a new level is inserted in a leveled binary tree only after the previous one has been fully populated, it is only the last level which shows that the leveled binary tree is not full – the sub-tree consisting of the root and all the levels up the last is a full binary tree. In other words, the tree of level $h = \lfloor \log 2n \rceil$ is not full, but its sub-tree of level h - 1 is full. This means that there are at least 2h-1 - 1 left edges in any leveled binary tree.

The maximum number of nodes in the last level of a leveled binary tree is 2h – when this number of nodes is reached, the leveled tree becomes a full tree. In order to keep the number of left edges to a minimum, only right child nodes would be inserted in the last level as much as possible. The number of possible right child nodes that can be inserted in the last level is 2h /2 = 2h-1 – after this many right child nodes are inserted, there is no more room for right child nodes in the last level, so the next node to be inserted would have to be a left child node, thus forcing the increase of the number of left edges in the leveled binary tree. As an example, the leveled binary tree in Fig. 1e has the minimum possible number of left edges, but the next node to be inserted in the last level has to be a left child node, as shown in Fig. 1f, so the number of left edges in the tree must increase.

In order for the number of left edges to be forced to increase, the number of nodes in the leveled binary tree must be greater than the number of nodes in all levels before the last plus half of all possible nodes in the last level. Thus, in order for the number of left edges to be forced to increase, it is necessary that $n > 2h - 1 + 2h-1$, so $n > 3 \cdot 2h-1 + 1$, thus $n - 3 \cdot 2h-1 + 1 > 0$. Using the Heaviside step function, it can be said that if $H(n - 3 \cdot 2h-1 + 1) = 1$ the minimum number of left edges in a leveled binary tree increases, whereas if $H(n - 3 \cdot 2h-1 + 1) = 0$ the minimum number of left edges in a leveled binary tree stays 2h-1 - 1.

The amount by which the minimum number of left edges increases in a leveled binary tree is the same amount required for the minimum number of left edges to increase. This means that the minimum number of left edges in a leveled binary tree will be $2h-1 - 1 + n - 3 \cdot 2h-1 + 1$ if $n - 3 \cdot 2h-1 + 1 > 0$ and 2h-1 - 1 otherwise. Employing the Heaviside step function notation, the expression $2h-1 - 1 + (n - 3 \cdot 2h-1 + 1) \cdot H(n - 3 \cdot 2h-1 + 1)$ is obtained, which is shown in equation (1).

3.3. Integer Sequence

Obtaining the results for progressive values of n using this formula gives the following values: 0, 0, 1, 1, 1, 2, 3, 3, 3, 3, 3, 4, 5, 6, 7, 7, 7, 7, 7, 7, 7, 7, 7, 8, 9, 10, 11, 12, 13, 14, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 31, 31, 31, 31, 31, 31, 31..., and this sequence can be expanded to infinity. It starts with the value for $n = 1$, as values for $n \leq 0$ are disregarded (since there are no binary trees with $n \leq 0$ nodes). The first nine values of this sequence correspond with the number of left edges in the leveled binary trees of Figure 1a to 1h. This sequence has been included into the On-Line Encyclopedia of Integer Sequences [19].

4. Maximum Number of Single-direction Edges in a Leveled Binary Tree

4.1. Motivation and Formula

Following the discovery of the integer sequence linked with the minimum number of single-direction edges in a leveled binary tree, curiosity prompted research about the regularity which governs the maximum number of singledirection edges in a leveled binary tree as well. Following the same approach, an algorithm was devised first and the formula was extracted afterwards. The formula is shown in (2).

$$a_{max}(n) = (n + 2^{h-1} - 1) \cdot He + (-1)^{He} \cdot (2^h - 1) \quad (2)$$

where, once again, $h = \lfloor \log 2n \rceil$ and represents the height of the leveled binary tree, and $He = H(-n + 3 \cdot 2h-1 - 1)$, where, again, H() is the Heaviside step function.

The proof will concern the maximum number of right edges in a leveled binary tree, to enable the use of Figure 1 again. The formula for the maximum number of left edges in a leveled binary tree is identical and its proof is analogous.

4.2. Proof of the Formula

Once again, h = ⌊log2n⌉ is the level of the leveled binary tree, 2h-1 - 1 is the number of single-direction (in this case, strictly right) edges in the full binary sub-tree of level h-1, and 2h is the maximum possible number of nodes in the last level (h) of the leveled binary tree. In it, it is possible to insert strictly right nodes, thus increasing the number of strictly right edges, up to a certain threshold, after which the maximum number of right edges remains unchanged until a next level is reached. The threshold is identical as in the previous formula, except that the increase of the number of right edges in the last level will take place while the threshold is not reached, and afterwards that number will remain unchanged until a next level is reached. Thus, it can be said that the number of right edges in the last level of the leveled binary tree will increase while $n < 2h - 1 + 2h\text{-}1$, i.e., $n < 3 \cdot 2h\text{-}1 - 1$, or, stated differently, $-n + 3 \cdot 2h\text{-}1 - 1 > 0$. Stated using the Heaviside step function, the number of strictly right edges will increase as long as H(-n + 3·2h-1 - 1) = 1, and will remain unchanged as long as H(-n + 3·2h-1 - 1) = 0. To shorten the writing, the annotation He = H(-n + 3·2h-1 - 1) is used.

The maximum number of right edges in a leveled binary tree is the number of right edges in the full binary sub-tree plus half of the possible edges in the last level, which equals to 2h-1 - 1 + 2h-1 = 2h - 1. On the other hand, if the last level is not completely filled with right edges, the total number of right edges will equal to the total number of edges in the tree minus the number of left edges in the full binary sub-tree of level h-1, which equals to n - 1 - (2h-1 - 1) = n - 2h-1. In other words, the maximum number of right edges in a leveled binary tree is n - 2h-1 when He = 1 and 2h - 1 when He = 0. Both cases can be included in a single expression when stated as (n + 2h-1 - 1)·He + (-1)He · (2h - 1), which represents equation (2).

4.3. Integer Sequence

Obtaining the results for progressive values of n using this formula gives the following values: 0, 1, 1, 2, 3, 3, 3, 4, 5, 6, 7, 7, 7, 7, 8, 9, 10, 11, 12, 13, 14, 15, 15, 15, 15, 15, 15, 15, 15, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 32, 33, 34, 35, 36, 37, 38..., and this sequence can also be expanded to infinity. It also starts with the value for n = 1, and the first nine values of this sequence correspond with the number of right edges in the leveled binary trees of Figure 1a to 1h. This sequence has also been included into the On-Line Encyclopedia of Integer Sequences [20].

5. Conclusion

In this paper, leveled binary trees are introduced as binary trees in which nodes are inserted in a breadth-first fashion. It is shown that special cases of the leveled binary trees are the complete binary trees and full binary trees. Inserting new nodes in a leveled binary tree while maintaining preference for a single direction, i.e., strictly left or strictly right child nodes, leads to the number of the single-direction edges increasing according to certain regularities. It is shown how the minimum and maximum numbers of single-direction edges in leveled binary trees are obtained according to certain formulae, which are presented and proved. Both formulae produce particular integer sequences which can be expanded to infinity, and both of those sequences have been included into the Online Encyclopedia of Integer Sequences.

The concept of a leveled binary tree is a novel one, and there are no previous results involving leveled binary trees as such, as far as the author could find. It can therefore be inferred that there is no previous research on this topic, and the hope of the author is that this paper will spark further such research and leveled binary trees will find their place and purpose in science. It can be said that the first such purpose is to be used as a data structure based on which integer sequences can be explained and generated, such as the two integer sequences presented in this paper, and hopefully there will be more.

Acknowledgement

George Tanev was an MSc student doing his Master's Thesis under the supervision of the author during the work shown in this paper. Thus, he contributed to it, especially to the first of the two sequences presented herein.

References

[1] Sedgewick, R. (1998). Algorithms in C++, Parts 1-4: Fundamentals, Data Structures, Sorting, Searching, 3rd ed, Addison-Wesley.

[2] Kreher, D. L., Stinson, D. R. (1998). Combinatorial Algorithms: Generation, Enumeration, and Search, *Discrete Math*

ematics and its Applications (Book 7), CRC Press, 1st Edition, 1998.

[3] Brassard, G., Bratley, P. (2002). Fundamentals of Algorithmics, Prentice Hall of India, 2002.

[4] Bozinovski, A., Bozinovski, S. (2004). N-Queens Pattern Generation: An Insight into Space Complexity of a Backtracking Algorithm, Proc. 3rd *Int. Symp. Information and Communication Technologies*, Las Vegas, Nevada, USA, 281- 286, 2004.

[5] Rich, E. (1983). Artificial Intelligence, McGraw-Hill series in artificial intelligence, McGraw-Hill Inc., 1983.

[6] Suh, I., Headrick, T. C. (2010). A Comparative Analysis of the Bootstrap Versus Traditional Statistical Procedures Applied to Digital Analysis Based on Benford's Law, *J. Forensic and Investigative Accounting*, 2 (2), p 144-175, 2010.

[7] Buchmann, J., Dahmen, E., Klintsevich, E., Okeya, K., Vuillaume, C. (2007). Merkle Signatures with Virtually Unlimited Signature Capacity, In: Proceedings 5th Int. Conf. Applied Cryptography and Network Security, Zhuhai, China, p 31-45, 2007.

[8] Katz, J. (2003). Binary Tree Encryption: Constructions and Applications, Information Security and Cryptology (ICISC 2003), vol. 2971, *Lecture Notes in Computer Science*, p 1-11, Springer, 2003.

[9] Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C. (2009). Introduction to Algorithms, 3rd ed., The MIT Press, 2009.

[10] Adel'son-Vel'skii, G. M., Landis, E.M. (1962). An Algorithm for the Organization of Information, Soviet Math. Doklady, vol. 3, p 1259-1263, 1962.

[11] Bayer, R., McCreight. (1972). Organization and Maintenance or Large Oriented Indexes, *Acta Inform*., 3 (3), p173- 189, 1972.

[12] Roch, S., Steel, M. (2014). Likelihood-Based Tree Reconstruction on a Concatenation of Alignments can be Statistically Inconsistent, *Theor. Popul. Biol*., vol. 100, p 56-62, 2014.

[13] Li, Y., Xu, M., Zhao, H., Huang, W. (2016). Hierarchical Fuzzy Entropy and Improved Support Vector Machine Based Binary Tree Approach for Rolling Bearing Fault Diagnosis, *Mech. Mach. Theory*, vol. 98, p 114-132, 2016.

[14] Liu, B., Shen, Y., Chen, X., Chen, Y., Wang, X. (2014). A partial binary tree DEA-DA cyclic classification model for decision makers in complex multi-attribute large-group interval-valued intuitionistic fuzzy decision-making problems, *Inf. Fusion*, vol. 18, p 119-130.

[15] Lee, Y., Lee, J. (2015). Binary tree optimization using genetic algorithm for multiclass support vector machine, *Expert Syst. Appl*, 42 (8), p 3843-3851, 2015.

[16] Amani, M., Lai, K.A., Tarjan. R. E. (2016). Amortized rotation cost in AVL trees, *Inf. Process*. *Lett*., 98 (5), p 327-330, 2016.

[17] Goodrich, M.T., Tamassia, R., Goldwasser, M. H. (2014). *Data Sructures and Algorithms in Java*, 6th ed., Wiley, 2014.

[18] Horowitz, E., Sahni, S. (1983). *Fundamentals of Data Structures. Computer Science* Press, 1983.

[19] Bozinovski, A., Tanev, G. (2016). Sequence A277267 in the On-Line Encyclopedia of Integer Sequences, 2016, (accessed October 13, 2016) Available from: https://oeis.org/A277267.

[20] Bozinovski, A. (2016). Sequence A279521 in the On-Line Encyclopedia of Integer Sequences, 2016, (accessed December 23, 2016) Available from: https://oeis.org/A279521.