Scientific Journal of Impact Factor (SJIF): 4.72
e-ISSN (O): 2348-4470
p-ISSN (P): 2348-6406

International Journal of Advance Engineering and Research Development

Volume 4, Issue 10, October -2017

A SURVEY ON DIFFERENT METHOD OF BALANCING A BINARY SEARCH TREE

B. Deepa[1], Reshmi. S[2], Kirthika. B[3]

[1, 2, 3] *Department of Information Technology, Sri Krishna Arts and Science College*

Abstract -*This paper empirically examines the various ways of balancing the binary search tree. Since we know how to construct a binary search tree the only thing left is to keep it balanced. Obviously, we will need to re-balance the tree on each insert and delete, which will make this data structure more difficult to maintain compared to non-balanced search trees, but searching into it will be significantly faster.*

Keywords *– Binary Search Tree, AVL Tree, LL Rotation, RR Rotation, Zig-Zag Rotation.*

I. INTRODUCTION

Binary search trees (BST) are a particular type of container (i.e.) data structures that store "items" (such as numbers, names etc.) in memory [1]. They allow fast lookup, addition and removal of items, and can be used to implement either dynamic sets of items or lookup tables that allow finding an item by its key [2].

II. BINARY SEARCH TREE

A Binary Search Tree (BST) is a tree in which all the nodes follow the below-mentioned properties. The left sub-tree of a node has a key less than or equal to its parent node's key. The right sub-tree of a node has a key greater than to its parent node's key [3] [1]. A self-balancing search tree is any node-based binary search tree that automatically keeps its height (maximal number of levels below the root) small in the face of arbitrary item insertions and deletions [4]. These structures provide efficient implementations for mutable ordered lists, and can be used for other abstract data structures such as associative arrays, priority queues and sets.

The red–black tree, which is a type of self-balancing binary search tree, was called symmetric binary B-tree and was renamed but can still be confused with the generic concept of self-balancing binary search tree because of the initials [5] [2]. The main disadvantage is that we should always implement a balanced binary search tree - AVL tree, Red-Black tree, Splay tree. Otherwise, the cost of operations may not be logarithmic and degenerate into a linear search on an array

3.1 AVL Tree

An AVL tree is another balanced binary search tree. Named after their inventors, Adelson-Velskii and Landis, they were the first dynamically balanced trees to be proposed, they are not perfectly balanced, but pairs of sub-trees differ in height by at most 1, maintaining an O (logn) search time [5] [1].

3.1.1 Example of an AVL search tree

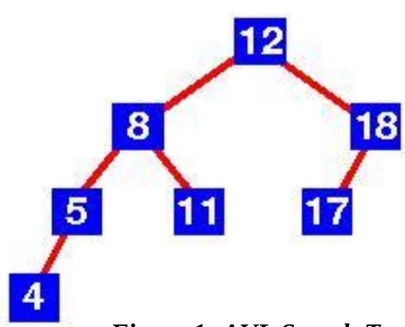

Figure 1: AVL Search Tree

Examination shows that each left sub-tree has a height 1 greater than each right sub-tree. Suppose a new key is inserted into this tree but once the new key is added we must update the balance factor of its parent if the balance factor is affected you can perform some kind of rotation to balance the avl tree according to the type of key is inserted [6].

3.1.2 AVL rotations

a. LL Rotation
b. RR Rotation
c. LR Rotation
d. RL Rotation

3. 2 Weight-balanced trees

A weight-balanced tree is a binary search tree that stores the sizes of subtrees in the nodes [6] [3]. That is, a node has fields

a. key, of any ordered type
b. value (optional, only for mappings)
c. left, right, pointer to node
d. Size of type integer.

By definition, the size of a leaf (typically represented by a nil pointer) is zero. The size of an internal node is the sum of sizes of its two children, plus one (size[n] = size[n.left] + size[n.right] + 1). Based on the size, one defines the weight as weight[n] = size[n] + 1 [7].

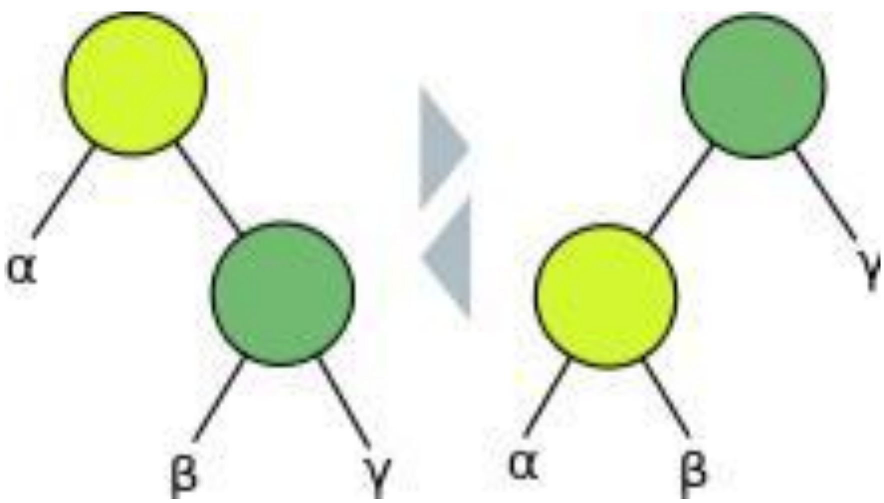

Figure 2: Weighted-Balanced Trees

3.2.1 Binary tree rotations

Operations that modify the tree must make sure that the weight of the left and right subtrees of every node remain within some factor α of each other, using the same rebalancing operations used in AVL trees: rotations and double rotations [8]. Formally, node balance is defined as follows:

node is α-weight-balanced if weight[n.left] ≥ α·weight[n] and weight[n.right] ≥ α·weight[n].[7]

Here, α is a numerical parameter to be determined when implementing weight balanced trees [9]. Larger values of α produce "more balanced" trees, but not all values of α are appropriate; Nievergelt and Reingold proved that.

3.3 A red–black tree

It is a kind of self-balancing binary search tree. Each node of the binary tree has an extra bit, and that bit is often interpreted as the color (red or black) of the node [10] [2]. These color bits are used to ensure the tree remains approximately balanced during insertions and deletions.

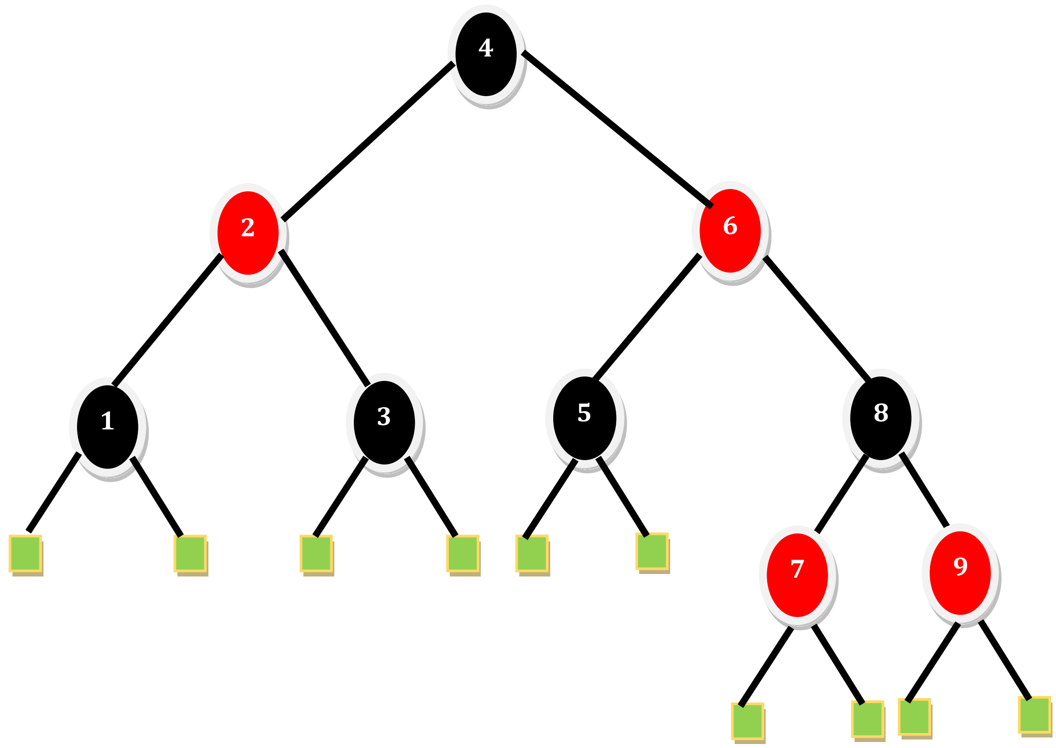

Figure 3: Red-Black Tree

3.4 Splay trees

Splay tree is another variant of binary search tree [11]. In a splay tree, the recently accessed element is placed at the root of the tree. A splay tree is defined as follows:

- Splay Tree is a self - adjusted Binary Search Tree in which every operation on an element rearrange the tree so that the element is placed at the root position of the tree.
- In a splay tree, every operation is performed at root of the tree. All the operations on a splay tree are involved with a common operation called "Splaying".
- Splaying an element is the process of bringing it to the root position by performing suitable rotation operations.
- In a splay tree, splaying an element rearrange all the elements in the tree so that splayed element is placed at root of the tree [12].

With the help of splaying an element we can bring most frequently used element closer to the root of the tree so that any operation on those elements performed quickly [13] [14]. That means the splaying operation automatically brings more frequently used elements closer to the root of the tree.

Every operation on a splay tree performs the splaying operation. For example, the insertion operation first inserts the new element as it inserted into the binary search tree, after insertion the newly inserted element is splayed so that it is placed at root of the tree [15] [16]. The search operation in a splay tree is search the element using binary search process then splay the searched element so that it placed at the root of the tree. In a splay tree, to splay any element we use the following rotation operations:

3.4.1 Rotations in Splay Tree

a. zig rotation
b. zag rotation
c. zig-zig rotation
d. zag-zag rotation
e. zig-zag rotation
f. zag-zig rotation

4 Conclusion

Although it is known that if input is random, we will be closer to a balanced tree [17] [18]. Still some balancing technique is required to prevent the tree from becoming higher on one side resulting after a series of insertions and deletions [19] [20]. Ultimate goal is to maintain the tree in such way that its height is always

O(lg(n)) so that all basic tree operations could be performed in O(lg(n)) time

5 References

[1] Adel'son-Vel'skii, G.M., and Landis, E.M., 1962, An Algorithm for the Organization of information. Soviet Mathematics Doklady, 3, pp.1259–1263.

[2] Day, A. C., 1976, Balancing a Binary Tree, Computer Journal, XIX, pp. 360-361.

[3] Martin, W.A., and Ness, D.N., Feb 1972, Optimal Binary Trees Grown with a Sorting Algorithm. Communication of the ACM 15, 2, pp. 88-93.

[4] M. Benaiah Deva Kumar and Deepa. B, "Computer Networking: A Survey", International Journal of Trend in Research and Development, IJTRD & ISSN 2394-9333, Volume 2(5), Sep-Oct 2015, pp – 126 – 130.

[5] Deepa. B, Srigayathri. S, Visalakshi. S, "A Review on Cloud Computing", International Journal of Trend in Research and Development, IJTRD & ISSN: 2394-9333, Volume 4(1), Jan-Feb 2017, pp - 197-199.

[6] Ira Nath and Dr. Rituparna Chaki, "BHAPSC: A New Black Hole Attack Prevention System in Clustered MANET", International Journal of Advanced Research in Computer Science and Software Engineering, 2(8), August 2012, ISSN: 2277 128X, Pg: 113-121.

[7] Reshmi. S, and M. Anand Kumar, "Survey on Identifying Packet Misbehavior in Network Virtualization", Indian Journal of Science and Technology, INDJST & ISSN (Online): 0974-5645, Vol 9; Issue 31, August 2016, Pg: 1-11.

[8] Reshmi. S, and M. Anand Kumar, "Secured Structural Design for Software Defined Data Center Networks", International Journal of Computer Science and Mobile Computing, IJCSMC & ISSN 2320–088X, IMPACT FACTOR: 5.258, Vol.5 Issue.6, June- 2016, pg. 532-537

[9] M. Anand Kumar, Dr. S. Karthikeyan (2011), "Security Model for TCP/IP Protocol Suite", Journal of Advances in Information Technology, 2[2], 87-91.

[10] M. Anand Kumar and Dr. S. Karthikeyan (2012)," Investigating the Efficiency of Blowfish and Rejindael (AES) Algorithms" International Journal of Computer Network and Information Security", 4[2]: 22-28

[11] M. Anand Kumar and Dr. S. Karthikeyan 2012)," A New 512 Bit Cipher - SF Block Cipher" International. Journal of Computer Network and Information Security", 4[11]:55-61.

[12] Dr. M. Anand Kumar.and Dr. S. Karthikeyan (2013)," An Enhanced Security for TCP/IP Protocol Suite" International Journal of Computer Science and Mobile Computing, 2[11]:331-338.

[13] Rajendra Aaseri, Pankaj Choudhary, and Nirmal Roberts, "Trust Value Algorithm: A Secure Approach Against Packet Drop Attack in Wireless Ad-Hoc Networks", International Journal of Network Security & Its Applications (IJNSA), 5(3), May 2013.

[14] Nishu Kalia, Harpreet Sharma, and Nishu Kalia, "Detection of Multiple Black hole nodes attack in MANET by modifying AODV protocol", International Journal on Computer Science and Engineering (IJCSE), ISSN: 0975-3397, 8(5) May 2016, pg 160 – 174.

[15] Manar Jammala, Taranpreet Singh, Abdallah Shami, RasoolAsal, Yiming Li, "Software-Defined Networking: State of the Art and Research Challenges", Elsevier's Journal of Computer Networks, October 2014, 72(1), Doi no: 10.1016/j.comnet.2014.07.004.

[16] Munoz-Arcentales Jose, Zambrano-Vite Sara, Marin-Garcia Ignacio, "Virtual Desktop Deployment in Middle Education and Community Centers Using Low-Cost Hardware", International Journal of Information and Education Technology, 2013 December, 3(6), Doi no: 10.7763/IJIET.2013.V3.355.

[17] Mohamed Ali Kaafar, Laurent Mathy, Thierry Turletti, Walid Dabbous, "Real attacks on virtual networks: Vivaldi out of tune", In Proceedings of the SIGCOMM workshop on Large Scale Attack Defense LSAD, 2006 September, 1(1), Doi no: 10.1145/1162666.1162672.

[18] J. Younge, R. Henschel, J. T. Brown, G. von Laszewski, "Analysis of Virtualization Technologies for High Performance Computing Environments", Cloud Computing (CLOUD), 2011 IEEE International Conference, 2011 July, 1(1), Doi no: 10.1109/CLOUD.2011.29.

[19] Ali Dorri and Hamed Nikde, "A new approach for detecting and eliminating cooperative black hole nodes in MANET", Information and Knowledge Technology (IKT), 7th Conference on IEEE, 2015.

[20] Pooja and Chauhan. R. K, "An assessment based approach to detect black hole attack in MANET", Computing, Communication & Automation (ICCCA), 2015 International Conference on. IEEE, 2015.