International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 3 Issue: 11 6200 - 6203

Binary Search Tree and Its Applications: A Survey

Mr. Chandrashekhar S. Khese
Department of Computer Engineering,
DGOI,FOE, Daund
Savitribai Phule Pune University,
Pune. India
chandrashekhar.khese@gmail.com

Prof. Amrit Priyadarshi
Department of Computer Engineering
DGOI, FOE, Daund
Savitribai Phule Pune University,
Pune, India
amritpriyadarshi@gmail.com

Abstract:- Binary search trees used as a data structure for rapid access to stored data. Arrays, vectors and linked lists data structures are limited by the trade-off between ability to perform fast search and resize easily. Complete and nearly complete binary search trees are of particularly significance. New version of insert-delete pair maintains random binary tree in a manner where all grandparents in tree always have both sub-trees full. In worst case binary search tree reduce to a linear link list, so reducing such search to sequential. In particular, we obtain a BST data structure that is O(log log n) competitive, satisfies the working set bound, dynamic ?nger bound and unified bound with an additive O(log log n) factor, and performs each access in worst-case O(log n) time.

*****

I. Introduction

The widespread adoption of multi-core processors places an in- creased focus on data structures that provide efficient and scalable multi-threaded access. These data structures are a fundamental building block of many parallel programs; even small improvements in these underlying algorithms can provide a large performance impact. One widely used data structure is an ordered map, which adds ordered iteration and range queries to the key-value association of a map. In-memory ordered maps are usually implemented as either skip lists or self-balancing binary search trees [1].

Binary search trees are used in computer science for rapid data storage and retrieval. With an ideally arranged BST with n nodes, most of the tree related operations require time that is not more than log(n). That means effort required to perform an operation on a BST grows logarithmically as size of the input grows. Despite of its wide popularity binary search tree has few serious problems. One of the major problems with binary search tree is its topology. The BST topology depends upon the order with which data is added or deleted. That means if input is not in random order the tree becomes lengthier on one side, reducing the search to be sequential. For optimal results the tree has to be wider and flatter in shape. In other words, tree height has to be minimal so that resulting tree could become bushy. To maintain the tree in better shape many algorithms have been proposed over the years. Some of them are [1], [2], [3], [5], and [6]. Since many versions of the insert-delete algorithms exist, we would like to touch upon the conventional insert-delete algorithms [2].

Binary search tree is most basic, nonlinear data structure in computer science that can be defined as “a finite set of nodes that is either empty or consists of a root and two disjoint subsets called left and right sub-trees. Binary trees are most widely used to implement binary search algorithm for the faster data access. When memory allocation is static and data size is reasonably small, an array may be used instead to accomplish the same task. However, for large data set array is not a good option since it requires contiguous memory that system may not provide sometimes. In ideal situation, we would expect the tree to be of minimal height that is possible only when the tree is height balanced [3].

One of the pillars of theoretical computer science is worst-case analysis: “assume the worst possible data.” By this account, binary search trees (BSTs) need O(log n) time for a search; in particular, information theory shows that searching1 for a uniformly random element requires O(log n) comparisons on average[4].

Electronic voting is a rising social application of cryptographic protocols. It promises the possibility of a convenient, efficient and secure facility for recording and tallying votes. Lot of literature on electronic voting has been developed over the last two decades. The uses of insecure Internet, incorrect implementations, and the resulting security breaches have caused substantial rework in this area. Several of these schemes were meant for secure electronic voting [5].

II. Related Work

An AVL tree [1] is a self-balancing binary search tree in which theheights of the left and right child branches of a node differ by nomore than one. If an insertion to or deletion from the tree causesthis balance condition to be violated then one or more rotations areperformed to restore the AVL invariant. In the classic presentation, nodes store only the difference between the left and right heights, which reduces storage and update costs. Balancing can also beperformed if each node stores its own height [1].

When analyzing the performance of a binary search tree. We require few parameters like its height or internal path length. Though, these parameters are interrelated but sometimes give better performance evaluation when analyzed independently. The height of the tree is the longest path from the root to a leaf node. The internal path length (IPL) of a tree is the sum of the depths of all nodes in the tree.

Root of the tree has depth zero, and every son in the tree has a depth that is one more than its parent. This means that IPL of the tree would be minimal when every node in the tree has minimal depth. This is possible only when the tree is height balanced. For an average case analysis of a binary search tree, the internal path length is an important parameter [2].

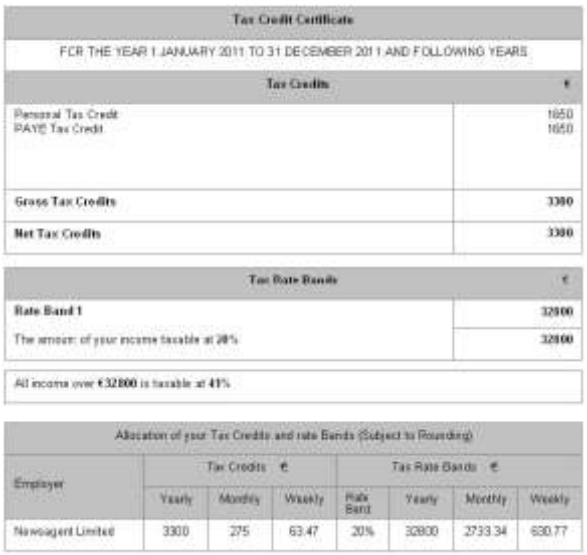
Soviet Mathematicians G. M.Adel'son-Vel'skii& E. M. Landis [1] proposed an algorithm to create a balanced binary search tree dynamically. Every node in the tree has to maintain additional information (apart from data and pointers) called "balance factor" that stores the effective balance of the tree rooted at that node. Tree is said to be balanced if the difference between the heights of two sub-trees of any node (balance factor) is between –1 and 1. Mathematically, –1 <= balance factor <=1. After each operation tree has to be examined to ensure that it is balanced. If the tree has become unbalanced appropriate rotation is performed in the appropriate direction [3].

As is standard in work on BST optimality, we consider only (successful) searches, not insertion and deletion. The letters n and m always refer to the size of a BST, and the total number of search operations performed on it, respectively. These are fixed global constants, and much of the notation depends on these values, either explicitly or implicitly. For simplicity, we denote the ordered values in the BST by the integers 1, 2,……,n[4].

The early investigations of e-voting used a simple voting approach. Boardroom voting protocol isan example [12]. Failure of a single voter can cause an election failure in this case. Simple votingschemes like Voter Verified Paper Audit Trial (VVPAT) and vote by mail are inherently insecureand provide no guarantee for the security or privacy [5].

III. Data Sources

a. Non-Medical Data:



Tax Credit Certificate

FOR THE YEAR 1 JANUARY 2011 TO 31 DECEMBER 2011 AND FOLLOWING YEARS

Tax Credits	€
Personal Tax Credit PAYE Tax Credit	1650 1650
Gross Tax Credits	3300
Net Tax Credits	3300

Tax Rate Bands	€
Rate Band 1	32800
The amount of your income taxable at 20%	32800

All income over €32800 is taxable at 41%

Allocation of your Tax Credits and rate Bands (Subject to Rounding)

Employer	Tax Credits €			Tax Rate Bands €			
	Yearly	Monthly	Weekly	Rate Band	Yearly	Monthly	Weekly
Newsagent Limited	3300	275	63.47	20%	32800	2733.34	630.77

Figure1. Tax Credit Certificate

Tax Credit certificate: If there is a change in your personal circumstances that affects the tax you pay, you need to tell Revenue. This could happen, for example, if you started work in a job where you can claim work-related expenses for a uniform. Revenue will then send you a new tax credit certificate that includes the changes. Revenue will give your employer the details needed to deduct the correct tax from your pay.

b. Medical Data:

Figure2. Medical Document

Medical data source are most important part in life insurance domain. As before applying for policy, individual needs to present medical health report along with all test details. Patient name, test date, addresses must include in report. Mainly bloods, ECG, kidney, bladder tests are common.

c. Policy Proposal:

Figure3. Policy Proposal Document

Genworth's life insurance forms will assist you with making changes to your index universal life, guarantee universal life, term life insurance and whole life insurance accounts. These forms will help you conduct life insurance

authorizations, requests, name/address change, naming a beneficiary and other updates [16].

d. Underwriter Data sources :

Figure4. Underwriter Document

Underwriting has been designed as an independent IT solution with a range of interfaces ensuring seamless integration with other systems operating in the insurance company, in particular, with the systems supporting the application processing, policy administration, claim management, as well as with a central register of insured and a document management system (DMS)[17].

IV. Applications

Binary Search Tree - Used in many search applications where data is constantly entering/leaving, such as the map and set objects in many languages' libraries.

Binary Space Partition - Used in almost every 3D video game to determine what objects need to be rendered. Binary space partitioning (BSP) is a method for recursively subdividing a space into convex sets by hyper planes. This subdivision gives rise to a representation of objects within the space by means of a tree data structure known as a BSP tree.

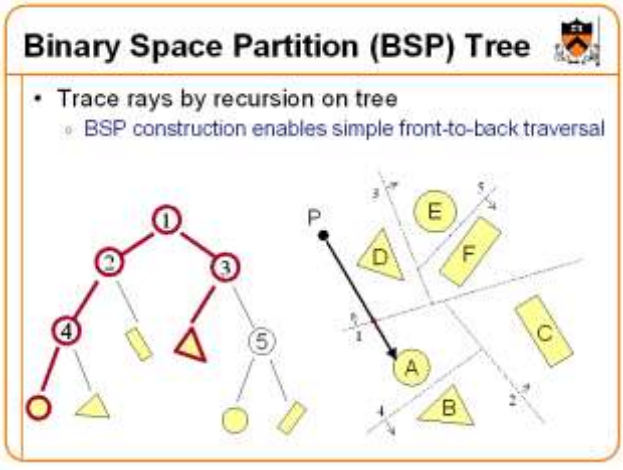


Figure5. BSP tree

Hash Trees - used in p2p programs and specialized image-signatures in which a hash needs to be verified, but the whole file is not available.

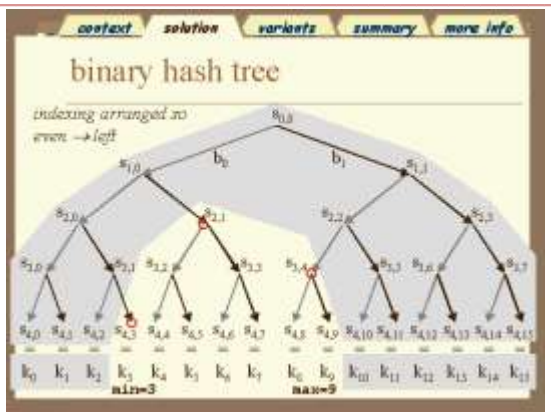


Figure6. Hash Tree

Huffman Coding Tree (Chip Uni) - Used in compression algorithms, such as those used by the .jpeg and .mp3 file-formats.

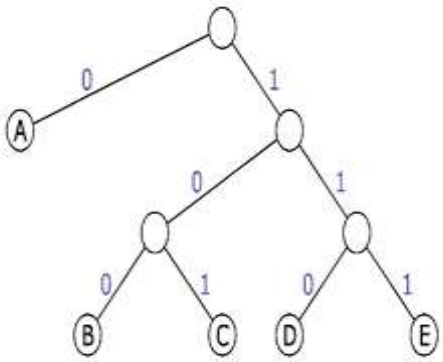


Figure7. Huffman Coding Tree

The branches of the tree represent the binary values 0 and 1 according to the rules for common prefix-free code trees. The path from the root tree to the corresponding leaf node defines the particular code word.

Syntax Tree - Parse trees are distinct from abstract syntax trees (also known simply as syntax trees), in that their structure and elements more concretely reflect the syntax of the input language.

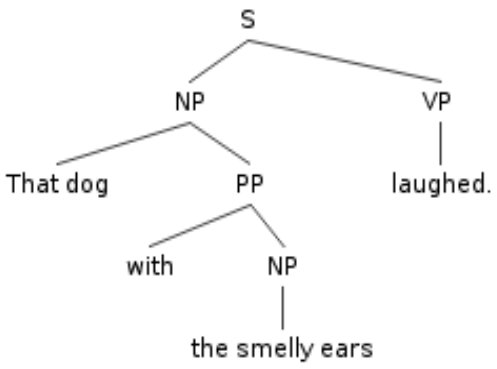


Figure8. Syntax Tree

V. Conclusion

Basically here we used different design variation which give support to fast clone operation. In case of random input balanced tree is always preferable. But still we need to focus on balancing techniques so that tree get prevent from becoming higher on one side. Main goal for tree operations is to perform task in O(log(n)) time. For this we maintain tree height always O(log(n)). Most algorithms are static and taking time linear to the input size and sometimes significant space amount is required. Static algorithms runtime overhead is less compared to dynamic algorithms. For balancing a tree in lesser time, algorithm is not yet developed. So there is huge improvement scope in existing methods.

Acknowledgement

I express great many thanks to Prof. AmritPriyadarshi and Prof. Sachin S. Bere for their great effort of supervising and leading me, to accomplish this fine work. To college and department staff, they were a great source of support and encouragement. To my friends and family, for their warm, kind encourages and loves. To every person who gave me something too light along my pathway. I thanks for believing in me.

References

[1] Nathan G. Bronson, Jared Casper, "*A Practical Concurrent Binary Search Tree*", Computer Systems LaboratoryStanford University, 2009.
[2] Jilani Abdul Khader, "*Insertion and Deletion on Binary Search Tree using ModifiedInsert Delete Pair: An Empirical Study*",Dept. of Computer Science, NizwaUniversity,Sultanate of Oman,2007.
[3] Suri Pushpa, Prasad Vinod, "*Binary Search Tree Balancing Methods: A Critical Study*", *Dept. of Computer Science and Applications, Kurukshetra University, Haryana, India*, 2007.
[4] Erik D. Demaine, Dion Harmon, "*The Geometry of Binary Search Trees*".
[5] Vinodu George, M P Sebastian, "*An Adaptive Indexed Binary Search Tree For Efficient Homomorphic Coercion Resistant Voting Scheme*", LBS College of Engineering, Kasaragod, Kerala, India ,2010.
[6] R. Sundar"*the deque conjecture for the splay algorithm*",Combinatorica, 1992.
[7] R. E. Tarjan "*Sequential access in play trees takes lineartime*". Combinatorica, 1985.
[8] C. C. Wang, J. Derryberry, and D. D. Sleator,"*competitive dynamic binary search trees*". SODA, 2006.
[9] L. Ballard, Conflict avoidance, "*Data structures in transactional memory*", Brown University Undergraduate Thesis, 2006.
[10] R. Bayer, M. Schkolnick, " *Concurrency of operations on B–Trees*", San Francisco, CA, USA, 1994.
[11] M. Herlihy, V. Luchangco, M. Moir, and W. N. Scherer, "*Softwaretransactional memory for dynamic-sized data structures*", New York, NY, USA,2003.
[12] K. S. Larsen, " *AVL trees with relaxed balance*", Washington, DC, USA, 1994.
[13] K. Fraser, "*Practical Lock Freedom*", University of Cambridge,2003.
[14] Vinod, P. Suri P, "*Maintaining aBinary Search Tree Dynamically. Proceedingsof the 10th International Conference onInformation Visualization*", London,UK,2006.
[15] J. Derryberry, D. D. Sleator, and C. C. Wang, "*A lowerbound framework for binary search trees with rotations*", Carnegie Mellon University, 2005.
[16] https://www.genworth.com/tools-and-forms/forms/life-insurance-forms.html
[17] http://insurance.comarch.com/articles/handling-a-life-insurance-underwriting-process

Authors

Mr. Chandrashekhar S. Khese. received his B.E. degree in Electronics Engineering from University of Mumbai in 2004. He has 10 years of experience of working with MNCs in Pune and Mumbai. He is currently working toward the M.E. Degree in Computer Engineering from University of Pune. His research interests lies in Data Mining, Software Engineering and Business Process Management.

Mr. AmritPriyadarshi received his B.E. degree in Electronics Engineering and MTech in Computer Science and Engineering. He has 10 years of experience as Assistant professor and he is currently perusing his Phd Degree.