THROUGH THE CONCEPT OF DATA STRUCTURES, THE SELF-BALANCING BINARY SEARCH TREE

Asniya Sadaf Syed Atiqur Rehman[1], Avantika kishor Bakale[2], Priyanka Patil[3]

[1,2,3]*Department of Computer Science and Engineering, Prof Ram Meghe College of Engineering and Management, Badnera*

ABSTRACT - *In the word of computing tree is a hierarchical data structure which stores information naturally in the form of hierarchy style. Tree is a most powerful and advanced data structure. This paper is mainly focused on self -balancing binary search tree(BST) also known as height balanced BST. A BST is a type of data structure that adjust itself to provide the consistent level of node access. This paper covers the different types of BST their analysis, complexity and application.*

Key words: AVL Tree, Splay Tree, Skip List, Red Black Tree, BST.

1. INTRODUCTION

Tree data structures are the similar like Maps and Sets, databases that quickly allows us to maintain a sorted list of numbers. Tree data structure is a hierarchical DS which stores the information in the form of hierarchy style and it provides an efficient operation like insertion and deletions operation. The introduction of tree by way of specific application consider the solutions on the problem of the performing a sequence of access operation. And a set of selected solutions from a totally ordered universes. In Tree DS different trees such as AVL, SPLAY, SKIP LIST, RED-BLACK has its own balancing methods and applications.

2. LITERATURE REVIEW

Soviet mathematician's G M Adelson and E M Landis (AVL) introduced an algorithm to maintain balance in a binary search tree with an additional property that every node in the tree has to maintain extra information (a part from data and pointer) known as balancing factor that contain a set of algorithms that allows multiple processor to concurrently update a step list is shared in memory this algorithm is very simple as compare to concurrent balancing tree algorithm. Which allow huge number of reader and many busy writers in skip list of n element with very little lock contention [1]. Skip list was invented by William Pugh in 1989. According to W. Pugh "skip list area probabilistic data structure that seen likely to supplant balanced tree as the implementation method of choice for the many apply. Skip list algorithm have same asymptotic expected time as balanced tree and are simple. Factor are useless apace." [2][3] Splay search tree was invented by Daniel Dominic sleator and Robert Tarjan in year 1985. Splay tree is a self-balancing binary search tree with extra property that recently accessed element is faster to access again. Red black tree is well-known way of implementing balancing tree as binary tree they are originally discovered by Bayer and are nowadays extensively (diseased) in the standard literature or algorithm Maintaining the invariant of red black tree through Haskell types system using the nested data types this give small list noticeable overhead. Many gives small list overhead can be removed by the use of existential types [4].

3.1. AVL TREE

AVL Tree is best example of self-balancing search tree; this means that AVL Tree is also binary search tree which is also balancing tree. A binary tree is said to be balanced if different between the height of left and right. Descendent in a tree is either -1, 0 or 1. In AVL tree every node maintain the extra information called as a balance factor. The formula for calculating the balance factor of AVL binary search tree is
Balance factor for AVL = height of left sub-tree – height of right sub-tree

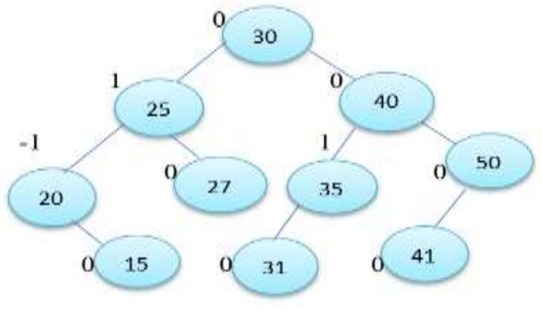

FIG -1: Diagram of AVL Tree

node satisfying the condition of the balance factor, therefore this tree is called as AVL Tree. Every AVL tree is binary search tree but all the binary search tree need not to be AVL tree. AVL tree is also known as height balancing tree. Consider an example- Let be non-empty binary tree with TL and TR as it left and right sub-tree. The height of tree is balance if and only if its satisfied the following condition.

I.TR and TL are height balance tree.
II.HL – HR <= 1 where HL is height of left sub-tree (TL) and right sub-tree (TR).

The balance factor of node of binary search tree can have value 1,0 -1depending on whether the height of sub-tree is greater less or equal to right sub-tree.

3.1.1 Advantages of AVL tree: Since AVL tree are self-balancing tree operation like deleting a node inserting a new node required the less time to be performed this directly reduced the complexity. Represent the AVL tree

```
struct AVL node
{
  int info;
  struct AVL node *left, *right;
 int balance-factor;
}
```

Algorithm for the different operation like insertion and deletion on the AVL

3.1.2 For insertion: Assumption-assume that tree is a binary search tree (BST)

Step1. Insert the element into tree using BST in logic
Step 2. check the balance factor for each node
Step3.if the balance factor for each node is either 0, 1, -1 then set it is AVL tree then print processed for the next operations.
Step4.exit
step5. else set imbalanced need to perform suitable rotation to perform to make balanced, then will processed for the next operation.
Step6.stop

3.1.3For deletion

Step1. Start the node where k is stored.
Step2.delete those content of the node
Step3. claim: deleting a node in an AVL tree can be reduced by leaf. three possible case: When X has no children then, delete X When X has one child, let X' become the child of X.
Step4.when X has two children, then find the X's successor Z
Step5.deleteZ

3.2 SPLAY SEARCH TREE

Splay search tree was invented by Daniel Dominic sleator and Robert Tarjan in year 1985.A Splay tree is another example of self-balancing binary search tree which have one special property that recently accessed element is quick to access again it perform same basic operation like any other BST such as insertion look up table and remove the 0(log n) amortized time. It is also used for the many sequence of non-random operation it also performs well than other search tree even when specific pattern of sequence is unknown.

All the operation which is performed on binary search tree are combined with one basic operation called splay in the term "splaying" is related to rearranging the element of C tree so the element is placed at the root of tree. The term splaying, we are tree rotation in a specific fashion to bring the element to top alternatively, a top-down algorithm can combine the search and recognize the tree into single phase [5].

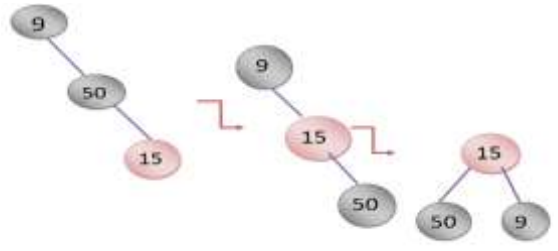

FIG -2: Diagram of Splay Tree

3.2.1 Advantages

The importance of the splay tree depends on the fact that it is self-optimizing in which recently access element is move root where they are quick to access again which is directly reduces the time require to access this reduce time complexity. This is very beneficial for many practical applications such as (locality of reference) and particularly useful for implementing cache and garbage collection algorithm. Advantage include Average case performance is efficient than other BST. Small memory to store any book keeping data.

3.2.2 Disadvantage

The most significant disadvantages of splay search tree are its height is liner for example –after accessing all n element in non-decreasing or order of tree. The time required is equal to worst case time. This means that actual cost of an actual cost of an operation is high however the "amortize" (in computer science amortize analysis is method for analysing the algorithm complexity 1) access cost of worst case is O (log n) also the expected access cost is can be reduced to O (log n) by using randomized variant [2]. The representation of splay tree can be changed even when they are accessed in read only manner this reduces its importance while using in purely functional programming due to this it also used in very limited ways to implement priority queue. It performs the 4 types of operation

1.Join
2.Split
3.Insertion

3.2.2.1Join: Assumption –assume that P and Q are the two BST, such that all the element of P Are Q them the following step can be used to for them use to a single tree. Splay tree largest it element in P this become the root of P has a no right child. Set right child of new root to Q.

3.2.2.2. Split: splay X now it is in the root tree to left contain all element smaller than X and tree to right contain all element larger them X.

3.2.2.3 Insert: To insert the new element in X. Insert X with a normal b. When element is inserted apply splay tree result newly inserted node is the root of tree.

3.3 SKIP LIST

In the world of computation quick search within an ordered sequence of element. Instant or fast search is only made possible by maintaining a linked hierarchy of sequence with each successive subsequence .skipping our fewer element their pervious one search start in the infrequent subsequence until two consecutive element have learn found one longer one smaller or one may equal to element searched for through this link hierarchy these two element link to next element of infrequent or sparest subsequence where searching is continued until finally we are searching in the complete sequence. The element is skipped over may be chosen probabilistically or deterministically with more common form. skip list is built in the form of layer. The bottom most layer is an ordinary order link list each top layer where an element in layer I appear in 1 in top most 1-p Element in the all the list. the skip list also includes log1\p of n list. A search for target element begin the head element in list and start processing horizontally until the current element is greater than or equal to target of current element is equal to target .it is found. if it is greater than target or search reach the end of link list the procedure is respected after recurring to previous element and dropped down vertically to the next list. the maximum number of step in each list is 1\p which can be seen by tracing the search path backword form the target until search at element that appear in the next higher list or reached to the current list. The maximum expected cost of search is log 1\p n which o (log n) Where p is any constant value. Through the implement of skip list. The element used for the skip list contain more than one pointer so they can easily participate in many list.

3.4 Red-black tree

The concept of each node of binary has an extra bit in the red-black tree that bits are offer interpreted as the red or black of the node the colour lists are used to ensure that the tree remain approximately balanced during insertion and deletion [6]. A red-black tree is self-balancing binary tree where each node satisfies the following condition are Every node has a colour either red or black. There should not be two adjacent red nodes (a red node cannot have a red parent or red child) Every path from root to null node has same number of black node. Need of red-black tree: many of the binary search tree, each operation taken 0(h) times where h is the height of BST the cost of these operations becomes 0(h) for skewed binary tree. It we ensure that height of tree remains 0(log n) after every operation then we can ensure an upper bound of 0(log n) for all these operations the height of red black is always 0(log n) where n is the number of node in BST. a red-black tree maintains a balance in the following manner: consider a simple example of understand balancing is a chain of three nodes is not possible in the red black tree. We can try any combination of colour and see all of them satisfy the red.

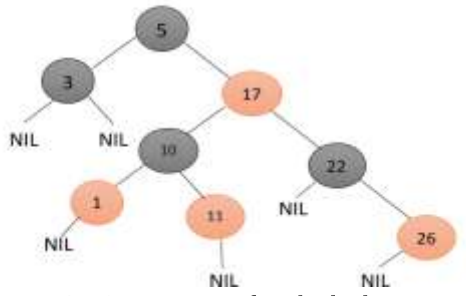

FIG – 3: Diagram of Red-Black Tree

4. CONCLUSION

In this paper we discussed about the various self - balancing binary tree their algorithm, complexity and application. The concept of self-balancing BST provide specific application consider the problem of the performing a sequence of access operation and set of selected solution from totally ordered universe. This BST is now considering experimental software for future work.

5. REFERANCE

[1] https://epaperpress.com/sortsearch/download/skiplist.pdf

[2] Pugh, W. (1990). "Skip lists: A probabilistic alternative to balanced trees" (PDF). Communications of the ACM. 33 (6): 668. doi:10.1145/78973.78977.

[3] Munro, J. Ian; Papadakis, Thomas; Sedgewick, Robert (1992). "Deterministic skip lists" (PDF). Proceedings of the third annual ACM-SIAM symposium on Discrete algorithms (SODA '92). Orlando, Florida, USA: Society for Industrial and Applied Mathematics, Philadelphia, PA, USA. pp. 367–375. doi:10.1145/139404.139478

[4] https://www.researchgate.net/publication/259413698_Red-black_trees_with_types

[5] https://en.wikipedia.org/wiki/Splay_tree

[6] Red black 1:Cormen, Thomas H.; Leiserson, CharlesE.; Rivest, Ronald L.; Stein, Clifford (2001). &quot;Red–***Black Trees &quot***;. Introduction to Algorithms (second ed.). MIT Press.pp. 273–301. ISBN 0-262-03293-7ss