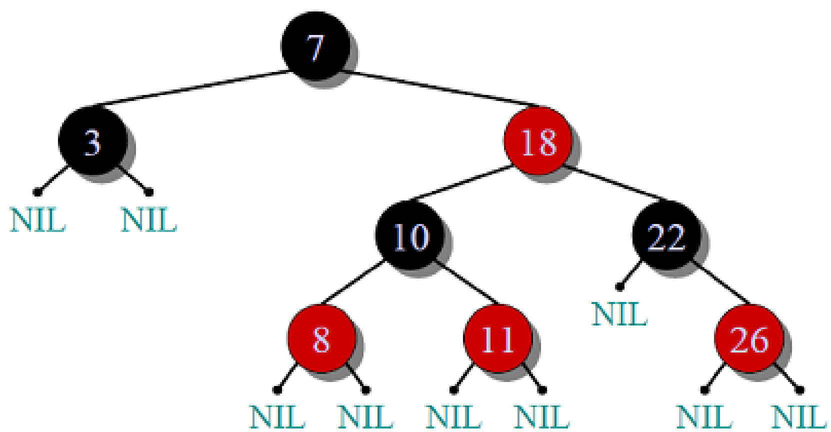
Self-Balancing-Binary-Search-Trees (Comparisons)

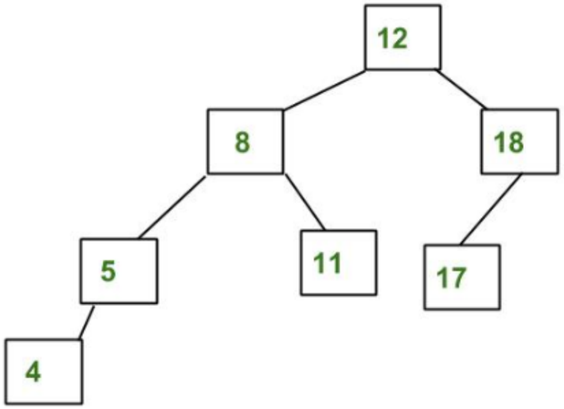
Self-Balancing Binary Search Trees are ***hei*** … utomatically keeps height as small as possible when insertion and deletion operations are p… y maintained in order of Log n so that all operations take O(Log n) time on average.

Examples :

Red Black Tree



AVL Tree:



How do Self-Balancing-Tree maintain height?

A typical operation done by trees is rotation. Following are two basic operations that can be performed to re-balance a BST without violating the BST property (keys(left) < key(root) < keys(right)). 1) Left Rotation 2) Right Rotation

```
T1, T2 and T3 are subtrees of the tree
rooted with y (on the left side) or x (on
the right side)
     y                               x
    / \     Right Rotation          /  \
   x   T3   - - - - - - - >        T1   y
  / \       < - - - - - - -            / \
 T1  T2     Left Rotation            T2  T3
Keys in both of the above trees follow the
following order
 keys(T1) < key(x) < keys(T2) < key(y) < keys(T3)
So BST property is not violated anywhere.
```

We have already discussed AVL tree, Red Black Tree and Splay Tree. In this acrticle, we will compare the efficiency of these trees:

METRIC	RB TREE	AVL TREE	SPLAY TREE
Insertion in worst case	**O(1)**	**O(logn)**	**Amortized O(logn)**
Maximum height of tree	**2*log(n)**	**1.44*log(n)**	**O(n)**
Search in worst case	**O(logn), Moderate**	**O(logn), Faster**	**Amortized O(logn), Slower**
Efficient Implementation requires	**Three pointers with color bit per node**	**Two pointers with balance factor per node**	**Only two pointers with no extra information**

METRIC	RB TREE	AVL TREE	SPLAY TREE
Deletion in worst case	O(logn)	O(logn)	Amortized O(logn)
Mostly used	As universal data structure	When frequent lookups are required	When same element is retrieved again and again
Real world Application	Multiset, Multimap, Map, Set, etc.	Database Transactions	Cache implementation, Garbage collection Algorithms

Recommended Posts:

Comparisons involved in Modified Quicksort Using Merge Sort Tree

Second minimum element using minimum comparisons

Counting Inversions using Ordered Set and GNU C++ PBDS

Shortest Path Faster Algorithm

Total number of BSTs using array elements

Ackermann Function

Count the number of words with given prefix using Trie

Vertical and Horizontal retrieval (MRT) on Tapes

Duplicates Removal in Array using BST

Build a segment tree for N-ary rooted tree

Minimum Cost Graph

Count of different groups using Graph

Count number of increasing sub-sequences : O(NlogN)

Median of sliding window in an array

Abhishek rajput
Check out this Author's contributed articles.

Article Tags : Advanced Data Structure Binary Search Tree AVL-Tree

Practice Tags : Binary Search Tree AVL-Tree

1

3.6

To-do Done

Based on **3** vote(s)

Feedback/ Suggest Improvement Add Notes Improve Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 201305
feedback@geeksforgeeks.org