AVL Tree | Set 1 (Insertion)

AVL tree is a self-balancing Binary Search Tree (BST) where the difference between heights of left and right subtrees cannot be more than one for all nodes.

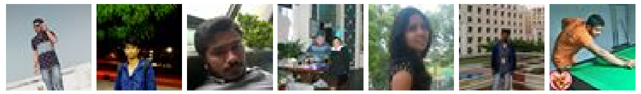
Why AVL Trees?

Most of the BST operations (e.g., search, max, min, insert, delete.. etc) take O(h) time where h is the height of the BST. The cost of these operations may become O(n) for a skewed Binary tree. If we make sure that height of the tree remains

O(Logn) for all these operations. The height of an AVL tree is always O(Logn) where n is the number of nodes in the tree (See this video lecture for proof).

Insertion

To make sure that the given tree remains AVL after every insertion, we must



the BST property (keys(left) < key(root) < keys(right)).1) Left Rotation2) Right Rotation

```
T1, T2 and T3 are subtrees of the tree rooted with y (on left si
de)
or x (on 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.
```

Steps to follow for insertion

Let the newly inserted node be w

1) Perform standard BST insert for w.

2) Starting from w, travel up and find the first unbalanced node. Let z be the first

grandchild of z that comes on the path from w to z.

3) Re-balance the tree by performing appropriate rotations on the subtree rooted with z. There can be 4 possible cases that needs to be handled as x, y and z can be arranged in 4 ways. Following are the possible 4 arrangements:

a) y is left child of z and x is left child of y (Left Left Case)

b) y is left child of z and x is right child of y (Left Right Case)

c) y is right child of z and x is right child of y (Right Right Case)

Popular Posts

- Top 10 Algorithms and Data Structures for Competitive

d) y is right child of z and x is left child of y (Right Left Case)

Following are the operations to be performed in above mentioned 4 cases. In all of the cases, we only need to re-balance the subtree rooted with z and the complete tree becomes balanced as the height of subtree (After appropriate rotations) rooted with z becomes same as it was before insertion. (See this video lecture for proof)

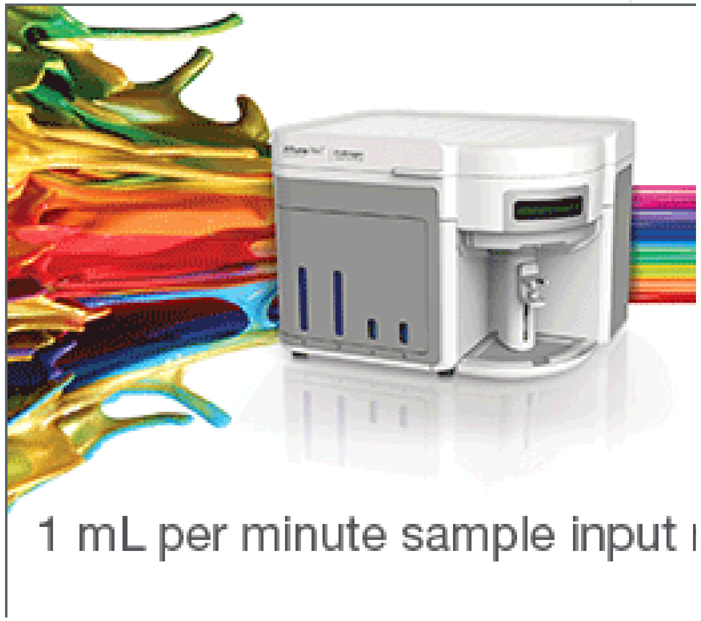
a) Left Left Case

```
T1, T2, T3 and T4 are subtrees.
         z                                      y
        / \                                   /   \
       y   T4      Right Rotate (z)          x      z
      / \          - - - - - - - - ->      /  \    /  \
     x   T3                               T1  T2  T3  T4
    / \
  T1   T2
```

b) Left Right Case

```
     z                               z
x
    / \                            /   \                        /
  \
   y   T4  Left Rotate (y)        x    T4  Right Rotate(z)    y
    z
  / \      - - - - - - - - ->    /  \      - - - - - - - ->  / \
   / \
T1   x                          y    T3                    T1  T2
 T3  T4
```

Programming
- Top 10 algorithms in Interview Questions
- How to begin with Competitive Programming?
- All permutations of a given string
- Memory Layout of C Programs
- Understanding “extern” keyword in C
- Heavy Light Decomposition
- Sorted Linked List to Balanced BST
- Comb Sort
- Aho-Corasick Algorithm for Pattern Searching



```
    / \                           / \
   T2   T3                       T1   T2
```

c) Right Right Case

```
   z                                y
  /  \                            /   \
 T1   y     Left Rotate(z)       z      x
     /  \   - - - - - - - ->    / \    / \
    T2   x                     T1  T2 T3  T4
        / \
      T3  T4
```

d) Right Left Case

```
    z                            z                            x
   / \                          / \                          /  \
 T1   y   Right Rotate (y)    T1   x      Left Rotate(z)   z
y
     / \  - - - - - - - - ->     /  \   - - - - - - - ->  / \    /
 \
    x   T4                     T2   y                  T1  T2  T3
   T4
   / \                                 /  \
 T2   T3                             T3   T4
```

implementation

uses the recursive BST insert to insert a new node. In the recursive BST insert, after

Learn more ›

invit by Thermo

Subscribe and Never Miss an Article

Email Address

Subscribe

- Common Interview Puzzles
- Interview Experiences
- Advanced Data Structures
- Dynamic Programming
- Greedy Algorithms
- Backtracking
- Pattern Searching
- Divide & Conquer
- Geometric Algorithms
- Searching
- Sorting
- Hashing
- Analysis of Algorithms
- Mathematical Algorithms
- Randomized Algorithms

insertion, we get pointers to all ancestors one by one in bottom up manner. So we don't need parent pointer to travel up. The recursive code itself travels up and visits all the ancestors of the newly inserted node.
1) Perform the normal BST insertion.
2) The current node must be one of the ancestors of the newly inserted node.
the height of the current node.
3) Get the balance factor (left subtree height – right subtree height) of the current node.
4) If balance factor is greater than 1, then the current node is unbalanced and we are either in Left Left case or left Right case. To check whether it is left left case or not, compare the newly inserted key with the key in left subtree root.
5) If balance factor is less than -1, then the current node is unbalanced and we are either in Right Right case or Right Left case. To check whether it is Right Right case or not, compare the newly inserted key with the key in right subtree root.

C Java

```
#include<stdio.h>
#include<stdlib.h>

// An AVL tree node
struct node
{
    int key;
    struct node *left;
    struct node *right;
```

- Recursion

All Categories

Select Category Update

Recent Comments

Ashish Jaiswal C code: #include<stdio.h> #include<stdlib.h>...
Detect Cycle in a Directed Graph · 55 minutes ago

Nikhil Ranjan I am interested in java.
Jobs · 1 hour ago

wilson yaa u r right
Find a Fixed Point in a given array · 1 hour ago

pranav adarsh C code for adjacency list representation of an...
Graph and its representations · 2 hours ago

akhil gupta public void boundryTraversal(Node<integer>...
Boundary Traversal of binary tree · 2 hours ago

```
    int height;
};

// A utility function to get maximum of two integers
int max(int a, int b);

// A utility function to get height of the tree
int height(struct node *N)
{
    if (N == NULL)
        return 0;
    return N->height;
}

// A utility function to get maximum of two integers
int max(int a, int b)
{
    return (a > b)? a : b;
}

/* Helper function that allocates a new node with the given
 key and
    NULL left and right pointers. */
struct node* newNode(int key)
{
    struct node* node = (struct node*)
                        malloc(sizeof(struct node));
    node->key   = key;
    node->left   = NULL;
    node->right  = NULL;
    node->height = 1;  // new node is initially added at le
```

Gitesh Narula

http://ideone.com/u3eOUs . Can anyone tell me...

Count Inversions in an array · 3 hours ago

Tags

Adobe Advance Data Structures Advanced Data Structures Amazon array Backtracking Bharti SoftBank (HIKE) Bit Magic C++ CN c puzzle D-E-Shaw DBMS Divide and Conquer Dynamic Programming Flipkart GATE GATE-CS-2012 GATE-CS-C-Language GATE-CS-DS-&-Algo GATE-CS-Older GFacts Goldman Sachs Google Graph Greedy Algorithm Hashing Interview Experience Java MAQ Software MathematicalAlgo Matrix Microsoft Morgan Stanley Operating systems Oracle Pattern Searching

```
af
    return(node);
}

// A utility function to right rotate subtree rooted with y
// See the diagram given above.
struct node *rightRotate(struct node *y)
{
    struct node *x = y->left;
    struct node *T2 = x->right;

    // Perform rotation
    x->right = y;
    y->left = T2;

    // Update heights
    y->height = max(height(y->left), height(y->right))+1;
    x->height = max(height(x->left), height(x->right))+1;

    // Return new root
    return x;
}

// A utility function to left rotate subtree rooted with x
// See the diagram given above.
struct node *leftRotate(struct node *x)
{
    struct node *y = x->right;
    struct node *T2 = y->left;

    // Perform rotation
    y->left = x;
    x->right = T2;

    //  Update heights
    x->height = max(height(x->left), height(x->right))+1;
    y->height = max(height(y->left), height(y->right))+1;

    // Return new root
    return y;
}

// Get Balance factor of node N
int getBalance(struct node *N)
{
    if (N == NULL)
        return 0;
    return height(N->left) - height(N->right);
}

struct node* insert(struct node* node, int key)
{
    /* 1.  Perform the normal BST rotation */
    if (node == NULL)
        return(newNode(key));

    if (key < node->key)
        node->left  = insert(node->left, key);
    else
        node->right = insert(node->right, key);

    /* 2. Update height of this ancestor node */
    node->height = max(height(node->left), height(node->rig
ht)) + 1;

    /* 3. Get the balance factor of this ancestor node to c
heck whether
       this node became unbalanced */
    int balance = getBalance(node);

    // If this node becomes unbalanced, then there are 4 ca
ses

    // Left Left Case
    if (balance > 1 && key < node->left->key)
        return rightRotate(node);

    // Right Right Case
    if (balance < -1 && key > node->right->key)
        return leftRotate(node);

    // Left Right Case
    if (balance > 1 && key > node->left->key)
    {
        node->left =  leftRotate(node->left);
        return rightRotate(node);
    }

    // Right Left Case
    if (balance < -1 && key < node->right->key)
    {
        node->right = rightRotate(node->right);
        return leftRotate(node);
    }

    /* return the (unchanged) node pointer */
    return node;
}

// A utility function to print preorder traversal of the tr
ee.
// The function also prints height of every node
void preOrder(struct node *root)
{
    if(root != NULL)
    {
        printf("%d ", root->key);
        preOrder(root->left);
        preOrder(root->right);
    }
}

/* Drier program to test above function*/
int main()
{
  struct node *root = NULL;

  /* Constructing tree given in the above figure */
  root = insert(root, 10);
  root = insert(root, 20);
  root = insert(root, 30);
  root = insert(root, 40);
  root = insert(root, 50);
  root = insert(root, 25);
  /* The constructed AVL Tree would be
            30
           /  \
         20   40
        /  \     \
       10  25    50
  */

  printf("Pre order traversal of the constructed AVL tree i
s \n");
  preOrder(root);

  return 0;
}
```

Output:

```
Pre order traversal of the constructed AVL tree is
30 20 10 25 40 50
```

Time Complexity: The rotation operations (left and right rotate) take constant time as only few pointers are being changed there.   Updating the height and getting the

same as BST insert which is O(h) where h is height of the tree.   Since AVL tree is balanced, the height is O(Logn). So time complexity of AVL insert is O(Logn).

The AVL tree and other self balancing search trees like Red Black are useful to get all basic operations done in O(Logn) time. The AVL trees are more balanced compared to Red Black Trees, but they may cause more rotations during insertion and deletion. So if your application involves many frequent insertions and deletions, then Red Black trees should be preferred. And if the insertions and deletions are less frequent and search is more frequent operation, then AVL tree should be preferred over Red Black Tree.

Following is the post for delete.
AVL Tree | Set 2 (Deletion)

Following are some previous posts that have used self-balancing search trees.

Median in a stream of integers (running integers)
Maximum of all subarrays of size k
Count smaller elements on right side

References:
IITD Video Lecture on AVL Tree Introduction
IITD Video Lecture on AVL Tree Insertion and Deletion

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

100 Comments Category: Advanced Data Structure Tags: Advance Data Structures

Related Posts:

- Centroid Decomposition of Tree
- Gomory-Hu Tree | Set 1 (Introduction)
- kasai's Algorithm for Construction of LCP array from Suffix Array
- Overview of Data Structures | Set 3 (Graph, Trie, Segment Tree and Suffix Tree)
- Heavy Light Decomposition | Set 2 (Implementation)
- Heavy Light Decomposition | Set 1 (Introduction)
- Count Inversions of size three in a give array

- Count inversions in an array | Set 3 (Using BIT)

Previous post in category | Next post in category

(Login to Rate and Mark)

3.8 Average Difficulty : **3.8/5.0**
Based on **5** vote(s)

🟩 Add to TODO List

🟩 Mark as DONE

63 people like this. Sign Up to see what your friends like.

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

100 Comments **GeeksforGeeks**

Recommend 8 Share

Join the discussion…

pranav adarsh • 16 days ago

IS This right code I am getting right result!!!!!

#include<stdio.h>

```
#include<malloc.h>

struct avl

{

int data;

struct avl*lf;

struct avl*rt;

int height;

};

int height(struct avl*p)
```

see more

• Reply • Share ›

Jay Solanki • a month ago

There is a problem with insertion.
If using this code if we try to insert 1->2->3
it becomes something like this

```
1
 \
  2
   \
```

3
Here the difference between height of left subtree and right sub tre
Ideally it should be

```
  2
 / \
1 3
```

But it is not happening using this code. as as per condition height
and as the left node is null the height of left subtree is 0. Other wis

see more

^ | ⌄ • Reply • Share ›

Naman Gupta • 2 months ago
Why there is a concept of LR and RL rotations? In and above prog
The second rotation call is made when the control returns to the p
balance factor is greater the 1)

^ | ⌄ • Reply • Share ›

Saurav Pradhan • 4 months ago
can avl tree for the same set be different??

^ | ⌄ • Reply • Share ›

Prabu R. • 4 months ago
https://github.com/Prabu073/pr...

^ | ⌄ • Reply • Share ›

Md Tareque Khan • 5 months ago

Can we replace the 4 if statement with if-else ladder in the insert function? I believe only atmost one of the four cases will match, s
if-else looks more correct. Besides, having four if, gives a wrong perception that the following if conditions can be true, if execution goes into the first if block (for example).

• Reply • Share ›

ganpat rao Md Tareque Khan • 5 months ago

bhakk saale.... ghanta bhi nahi aata tere ko :D

• Reply • Share ›

Vivek • 8 months ago

what will happen when insert function gets called with duplicate ke

• Reply • Share ›

danish-shark Vivek • 8 months ago

In the insert function, replace if (key < node->key) with if (k
strictly speaking AVL tree is a type of dictionary and all ent
keys. This condition is mandatory for a (key,value) pair to u

• Reply • Share ›

Monica Shankar danish-shark • 7 months ago

no you need a separate case to handle when value

1 • Reply • Share ›

Arya ➦ Vivek • 8 months ago

Tagda yaar

^ | ⌄ • Reply • Share ›

Vivek • 8 months ago

/* 1. Perform the normal BST rotation */

is it rotation or insertion??

^ | ⌄ • Reply • Share ›

Monica Shankar ➦ Vivek • 7 months ago

ordinary insertion where based on the values you traverse

^ | ⌄ • Reply • Share ›

Dman • 9 months ago

In insertion in AVL, is their maximum one rotation?

^ | ⌄ • Reply • Share ›

Dman • 9 months ago

simple implementation

https://ideone.com/7sYZnt

2 ^ | ⌄ • Reply • Share ›

nits • a year ago

should height be always incremented, there can be an insertion in ancestor.

Reply • Share ›

Monica Shankar ➦ nits • 7 months ago

Although not discussed here, the height won't be incremer

Reply • Share ›

John ➦ nits • 9 months ago

It is not necessarily incremented every time. The +1 is sim
children's height does not change, the current node's heig
height is ALWAY +1 the height of the children. Hope this h

Reply • Share ›

Piyush • a year ago

Here is my iterative solution :
http://ideone.com/oyMybx

Reply • Share ›

Varun Khare ➦ Piyush • 9 months ago

it is wrong

Reply • Share ›

Mission Peace • a year ago

https://youtu.be/rbg7Qf8GkQ4

Check out my video on avl tree.

Reply • Share ›

Guest • a year ago

consider :

z
/ \
y T4
/ \
x T3

now insert T1/T2 :

z
/ \
y T4
/ \
x T3
/
T1

So, imbalance is at z .... so instead of seeing key < node->left->ke
less than y , shouldn't we see key < node->left->left->key,
i.e. if T1 < x, or T2 >x ti distinguish between left left and left right c

we DON'T need to see for T3 because tree (z) will already be bala

Can anybody clarify it?

• Reply • Share ›

Sushil Lakra Guest • 9 months ago

This is left left imbalance, right rotate at z, y will become n

New tree will look something like this:

y
/ \
x z
/ / \
T1 T3 T4

Reply • Share ›

Right name • a year ago

can someone help me with some ideas for deliting a node in an A

Reply • Share ›

Abhinav Rai Right name • a year ago

Deleting the node in AVL tree is just the same as deleting t
also have to balance the tree; starting from the deleted noc

1 • Reply • Share ›

Right name Abhinav Rai • a year ago

Hello! I don`t know if you could give me some help.
Write the
code to transform a graph represented in an adjac
representation (and the other way around).
Also make sure that the BFS and DFS traversals a
new representation.

I could use some ideas here! Thanks:)

Reply • Share ›

Right name Abhinav Rai • a year ago

I`ll try. Thank you!

Reply • Share ›

Abhishek Singh • a year ago

This is incorrect implementation of AVL Tree. To be specific, getB
doesn't take care of case where
`(height(n->right) - height(n->left)) >=0` . As a result what we get i
pre-order traversal.

Reply • Share ›

GeeksforGeeks Mod Abhishek Singh • a year ago

Please take a closer look at the code. getBalance() is sup
conditions in insert() which use the returned balance to do

1 • Reply • Share ›

Evan Urena Abhishek Singh • a year ago

Note that AVL trees do not have to have leaf nodes that are
to have skewed AVL trees.

Reply • Share ›

Abhishek Singh Evan Urena • a year ago

I think the whole point of balanced trees like AVL/RI
complexity. If you're getting skewed trees in AVL, tl
any other non-balancing tree?

Reply • Share ›

Sumit Kesarwani • a year ago

@

• Reply • Share ›

Hello_world • a year ago

Is it possible to write AVL insertion code in iterative way ?

• Reply • Share ›

Biphph Hello_world • a month ago

Iterative AVL is only possible with Parent nodes. The soluti

• Reply • Share ›

danish-shark Hello_world • 8 months ago

Yes, I have implemented it in java. But trust me, the recurs

• Reply • Share ›

ganglu • a year ago

there is a bug in the code, for checks like

if (balance > 1 && key < node->left->key)

node->left could be null and so node->left->key could throw a null

consider this example

construct a new binary tree and insert the following keys in this or
[2,4,7]

1 ^ | v • Reply • Share ›

Sohaib ganglu • a year ago

This wont ever happen cause balance > 1 can only happe

1 ^ | v • Reply • Share ›

Jimmy Sohaib • a year ago

As 7 is inserted, the tree is
2 : height = 3
\
4 : height = 2
\
7 : height = 1

When 2 checks its balance it will be = 2
It will then check node.left.key and fire a null pointe
though since it's a very simple fix. Just add node.le
statements

^ | v • Reply • Share ›

Mohamed Ramdan Jimmy • a year ago

when 2 checks it will be -2 not 2 !

^ | v • Reply • Share ›

Jimmy Sohaib • a year ago

No, ganglu is correct. Try it yourself.

| • Reply • Share ›

petros • a year ago

There is a big memory leak in this code but I can't find it.
please help me.

• Reply • Share ›

am • a year ago

There is a bug somewhere in this code, if you insert from 1->64 it

• Reply • Share ›

helper • a year ago

my avl tree code, almost similar and in c http://ideone.com/Ge9y0

• Reply • Share ›

Jerry Goyal • a year ago

shouldn't it be x instead of w in "1) Perform standard BST insert fc

• Reply • Share ›

Guest • a year ago

@GeeksforGeeks in left and right rotation we are taking maximu
while updating height
y->height = max(height(y->left), height(y->right))+1;
x->height = max(height(x->left), height(x->right))+1;
i think it should be minimum

• Reply • Share ›

Optimus • a year ago

I think code is incorrect, because we are performing rotation (cons
code we are passing current node which is X. So how it will perfo
attached..As insertion always take place in the form of BST, so ins
performed in the current node instead of Z

^ | ⌄ • Reply • Share ›

Yitao Li • 2 years ago

Fantastic!
Except for some minor fix is actually needed to handle duplicate ir

```
*** before_dupl_fix.cc Sun Sep 14 17:08:34 2014
--- after_dupl_fix.cc Sun Sep 14 17:09:33 2014
***************
*** 94,99 ****
--- 94,100 ----

if (key < node->key)
node->left = insert(node->left, key);
+ else if (key == node->key) return node;
else
node->right = insert(node->right, key);
```

# This would be a test case that terminates with signal 11 before t

# struct node *root = NULL;

# root = insert(root, 1);
# root = insert(root, 0);

# root = insert(root, 1);
# root = insert(root, 1);
# root = insert(root, 1);
# root = insert(root, 0);

2 ^ | ⌄ • Reply • Share ›

Shekhar • 2 years ago

Height of Node :

return -1 ; if node ==NULL

^ | ⌄ • Reply • Share ›

sandeep • 2 years ago

in the "steps to follow insertion"
after standard BST insert shouldn't we update the balance factors

^ | ⌄ • Reply • Share ›

SubZero • 2 years ago

Wrong code Pal. Leaf node height is 0 and null is -1. All wrong info

^ | ⌄ • Reply • Share ›

Load more comments