Tuesday, 16 April 2024 9:28 PM

Binary Search Tree

Search (40)

TC → O(h)

log n

36 / 70, 15, 20, 55, 4, 6, 45

Right skewed tree

TC → O(h)

↓

O(n)

AVL : self balancing binary search tree

If : { -1, 0, 1 }

↳ if { $H_L - H_R$ }

BST

✗ AVL

If problem :

1) Left - left case

→ ① Insert (5)
② find that unbalanced node.
③

Left - left → Right Rotation

2) Right - Right case

① Insert 90
②
③

R R → L rotation

3) LR rotation

① Insert 30
②
③

L → stop

rotate 10 left

4) RL rotation

① Insert 90
②
③

30 20 25 30 10 5 15 27

Q) Given the roots of BST, root 1 and root 2, return true if and only if there is a node in the first tree and a node in the second tree whose values sum up to a given integer target.

Eg.

target → 5

output → true

target → 10

target → 18

8 → 5 5 5 1

1) Traverse tree ① & ②

a)

In order → L-Node

TC → O(n + m)

→ Iterative Inorder traversal

List →

Stack st = new Stack();

while(!, s : !empty () || root != null) {

while (root != null) {

s. push(root);

root = root. left;

}

root = s. pop();

if(data. compare(root. val) return true;

root = root. right;

}

8) Serialize and Deserialize Binary tree

Binary tree

BFS - level order

Serialize

[1, 2, 5, 3, 7, null, null]

Preorder

[1, 2, 3, null, null, 7, null, null, 5]

null, null

class node {

int val;

Node left;

node right;

}

String serialize (root) {

}

node String deserialize (String)

}

serialize (Node root; String str)

if(root == null) {

str += "null";

return;

} else {

str += str. valueOf(root. val) + ",";

str += serialize (root. left, str);

str += serialize (root. right, str);

}

return str;

}

deserialize (String data) {

String[] str = data. split (",");

Queue<Integer> q = new LinkedList<Integer>();

List<String> list = new LinkedList<String>(Arrays. asList(str));

de (q, list) {

if (list. get (0). equals (null) {

list. remove (0);

return null;

}

Node root = new Node (Integer. valueOf(list. get(0));

list. remove (0);

root. left = de(q, list);

root. right = de(q, list);

return root;

}