Binary Search Trees

BST is a binary tree in which for every node -

(i) left subtree contains the values lesser than the node's value
(ii) right subtree contains the values greater than the node's value.

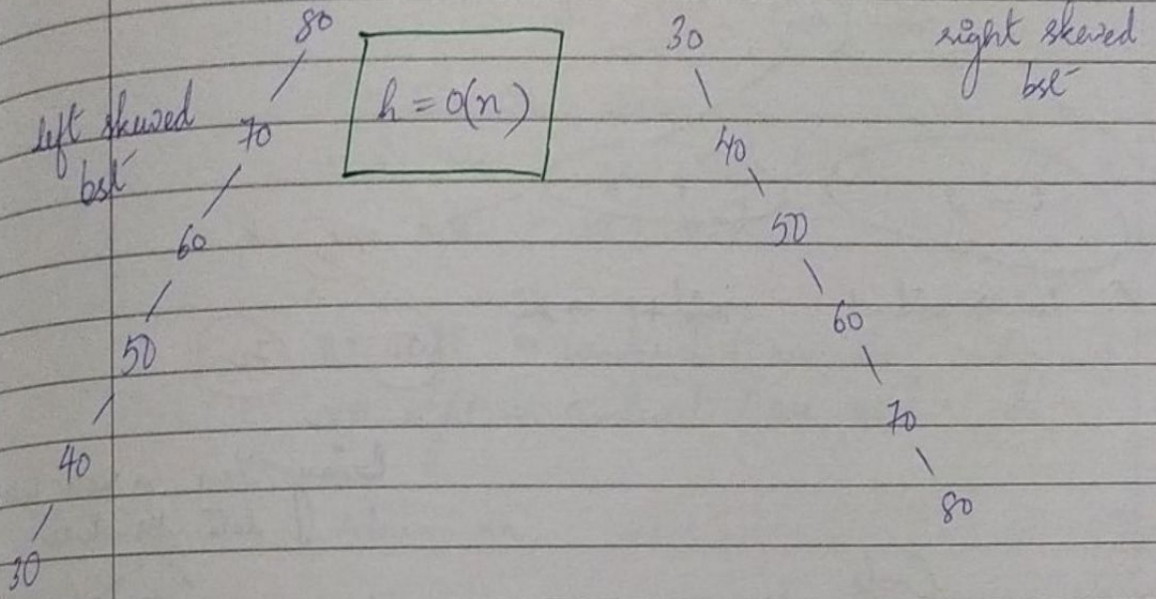
Properties:

PAGE NO.
DATE : / /

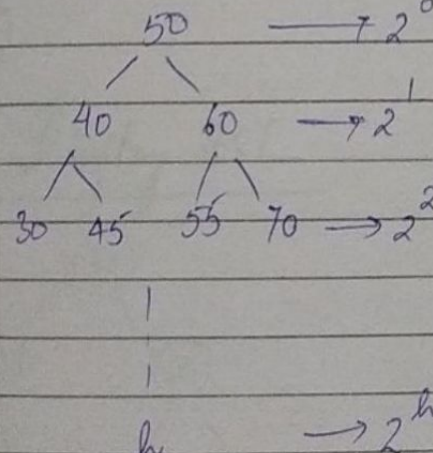
LNR

1) Inorder traversal will always be sorted.

2) left skewed tree / right skewed tree



Balanced bst



$2^0 + 2^1 + 2^2 + \cdots + 2^h = n$

$1\left(\frac{2^{h+1}-1}{2-1}\right) = n$

$2^{h+1} = n+1$

$h+1 = \log_2(n+1)$

$h = \log_2(n+1) - 1$

$\frac{a(r^n-1)}{r-1}$

$O(\log_2 n)$

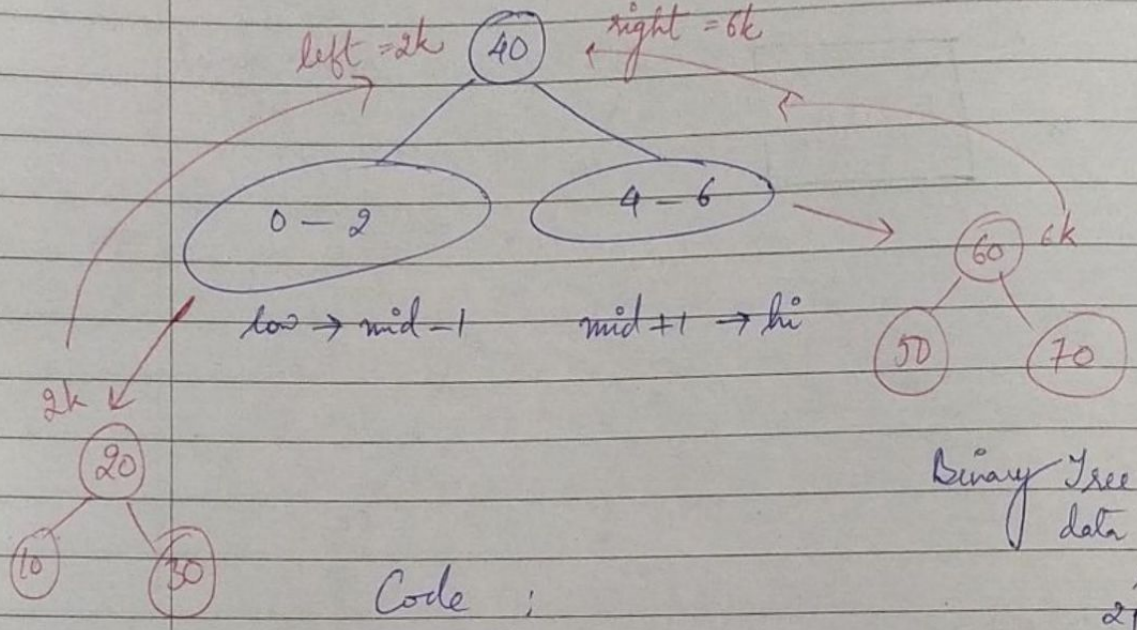
isko root banane se balanced tree banega

PAGE NO. icon
DATE : / /

lo → [10, 20, 30, 40, 50, 60, 70] ← hi

L N R

lo = 0
hi = 6
mid = (0+6)/2 = 3



Binary Tree → non linear data structure

2 paths hote hai to move.

Code :

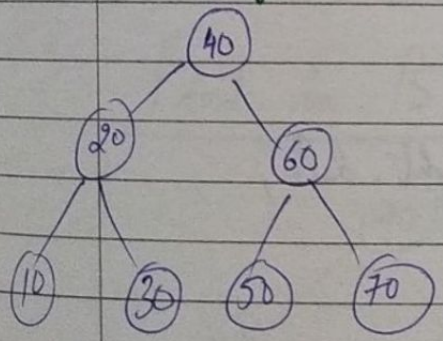
```
public class BST {

    private class Node {
        int data;
        Node left;
        Node right;
    }
    private Node root;

    public BST(int[] arr) {
        this.root = construct(arr, 0, arr.length - 1);
    }
```

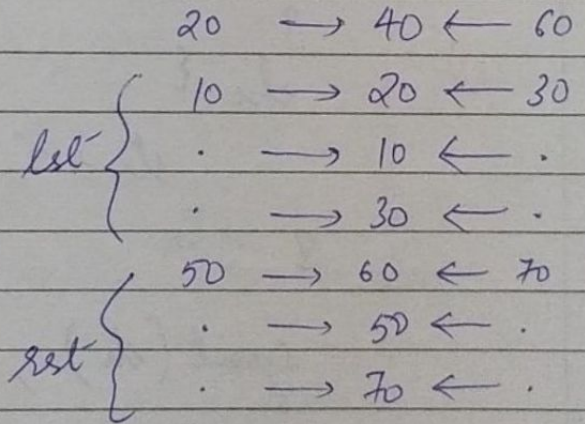
PAGE NO. icon
DATE : / /

Node

```
private void construct(int[] arr, int lo, int hi){
    BC  if(lo > hi){ return null}  ;
    // mid
    int mid = (lo + hi)/2 ;

    // create a new node
    Node nn = new Node ();
    nn.data = arr[ mid ];

    nn.left = construct( arr, lo, mid - 1 );
    nn.right = construct( arr, mid + 1, hi );

    return nn;

  }
}
```

Display BST





```
public void display () {
    display (this.root);
}

private void display ( Node node ) {
    BC  if (node == null)
            return

    // self work
    String str = "";
    if (node.left == null) {
        str += ".";
    }
    else {
        str += node.left.data;
    }

    str += "->" + node.data + "<-";

    if (node.right == null) {
        str += ".";
    }
    else {
        str += node.right.data;
    }

    sysout (str);
```

create n this A A A

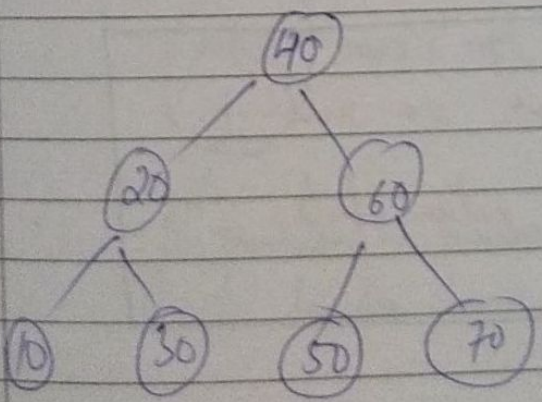
left

rst

* Code: Complete BST Implementation Github Repo

```
    // left
    display(node.left);
    // right
    display(node.right);
}

// Main

psvm(String[] args) {
    int[] arr = {10, 20, 30, 40, 50, 60, 70};
    BST bst = new BST(arr);
    bst.display();
}
```

Find in BST



item = 80

int item

```
public ~~void~~ boolean find () {
    return find (this.root, item);
}

private boolean find (Node node, int item) {
    BC
    if (item > node.data) {          // smaller problem
        return find (node.right, item);
    }
    else if (item < node.data) {     // smaller problem
        return find (node.left, item);
    }
    else
        return true;                 // self work
}
```

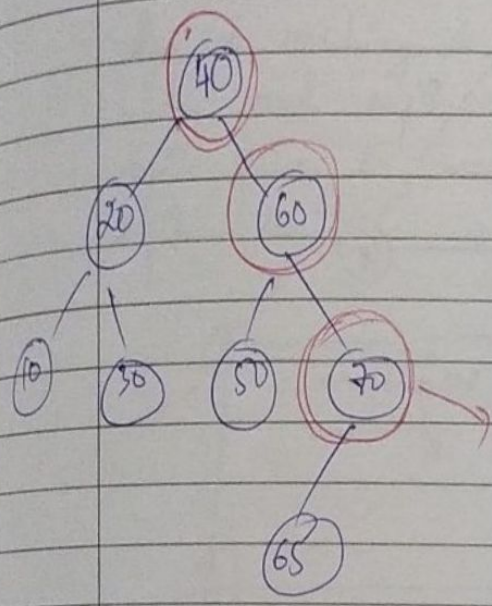
Recursion ka answer hi hamara answer hai toh return karenge

3.

BC → if (root == null)
        return false;

PAGE NO. icon
DATE : / /

Maximum in BST



① node.left == null &&
node.right == null

② node.left != null &&
node.right == null

Code :

```
private int max (Node node) {
    if (node.right == null) {
        return node.data;
    }
    return max (node.right);
}
```

3

jo recursion ka answer hoga
voh hamara answer hoga

↓

isliye return kardiya.