10 9 8 7 6 5 4 3 2 1

K = 5

10 8 13 24 19 18 9

0 K-1

K = 3

0 1 1 2

8 9 10 13 18 19 24

1 2

Example 1:

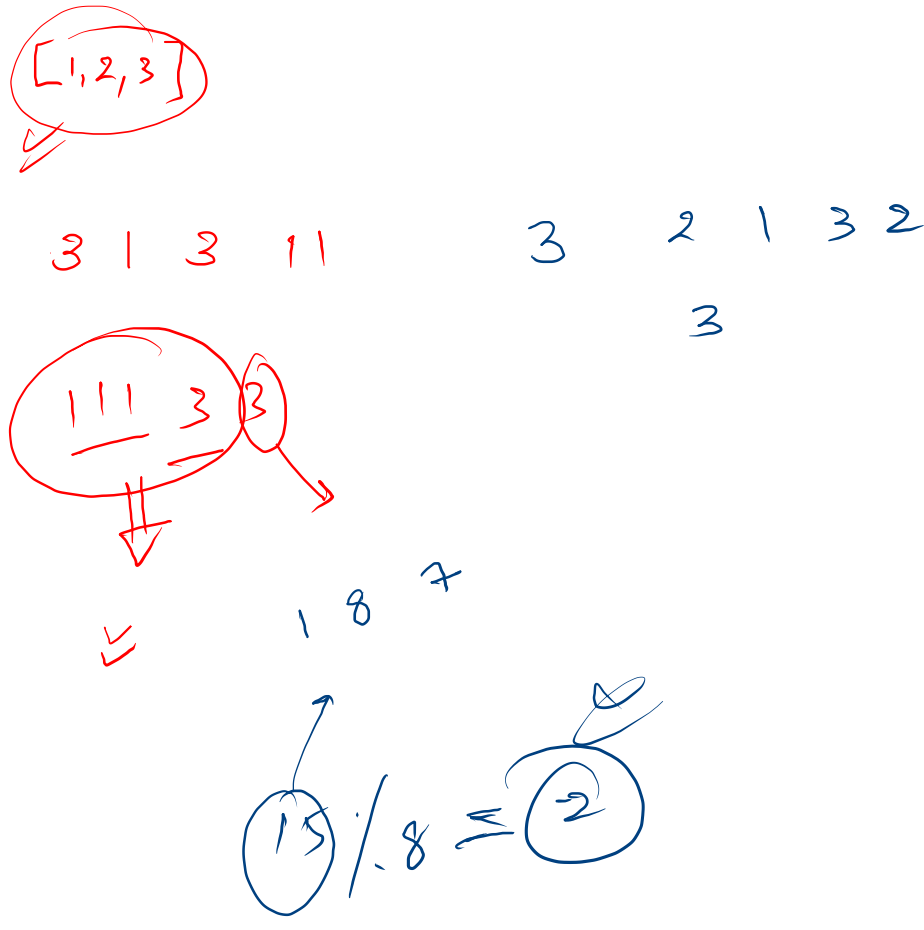
```
Input: tasks = [1,2,3], sessionTime = 3
Output: 2
Explanation: You can finish the tasks in two work sessions.
- First work session: finish the first and the second tasks in 1 + 2 = 3 hours.
- Second work session: finish the third task in 3 hours.
```

Example 2:

```
Input: tasks = [3,1,3,1,1], sessionTime = 8
Output: 2
Explanation: You can finish the tasks in two work sessions.
- First work session: finish all the tasks except the last one in 3 + 1 + 3 + 1 = 8 hours.
- Second work session: finish the last task in 1 hour.
```

Example 3:

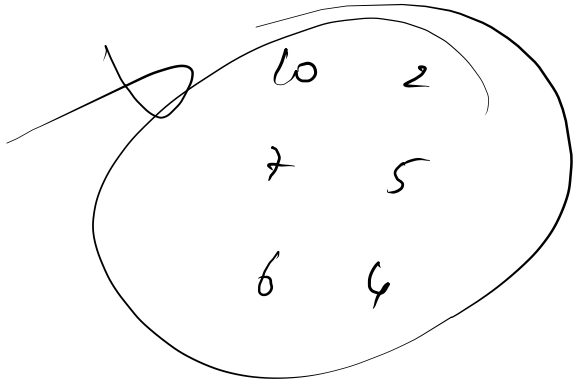
```
Input: tasks = [1,2,3,4,5], sessionTime = 15
Output: 1
Explanation: You can finish all the tasks in one work session.
```



[1,2,3,4,5]

[1,5,7,10,3,8,4,2,6,2]

10

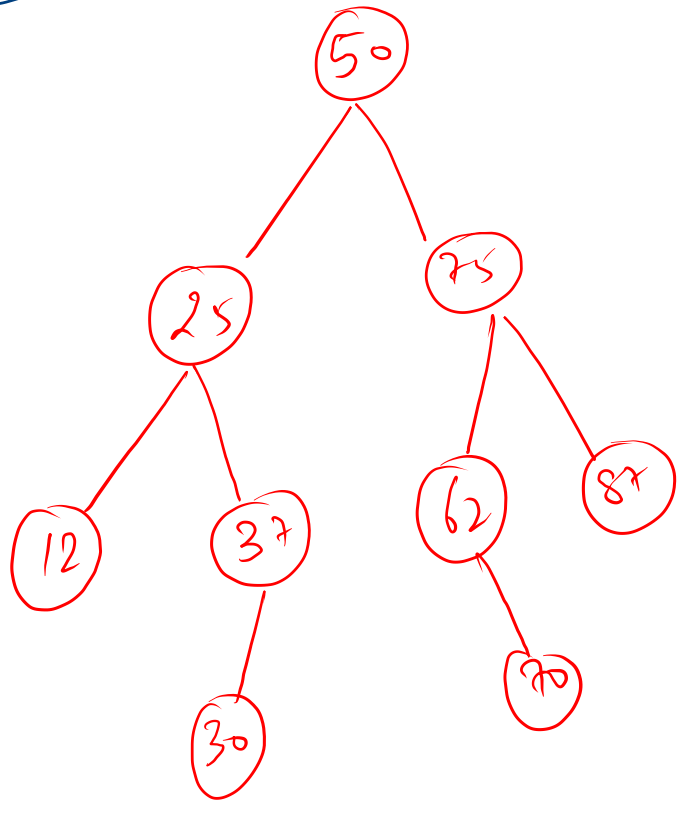


[2,3,3,4,4,4,5,6,7,10]
12

19
50 25 12 n n 37 30 n n n 75 62 n 70 n n 87 n n

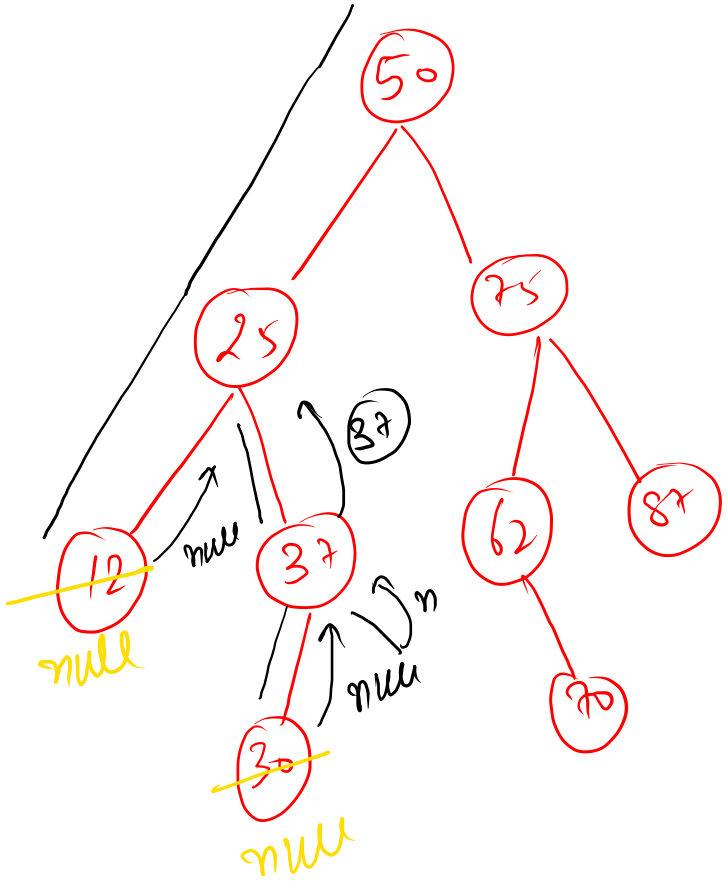
Remove Leaves

Input




```
public static Node removeLeaves(Node node){
    if(node == null){
        return null;
    }
    if(node.left == null && node.right == null){
        return null;
    }
    node.left = removeLeaves(node.left);
    node.right = removeLeaves(node.right);
    return node;
}
```

Output



19
50 25 12 n n 37 30 n n n 75 62 n 70 n n 87 n n

Generic Tree

Diameter

→ maximum distance b/w two nodes

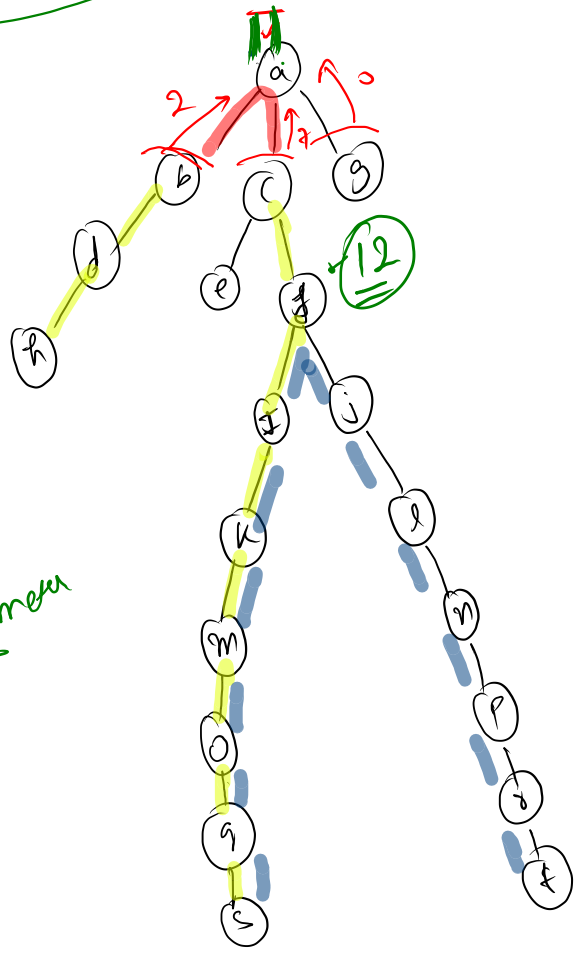
→ Edges

Node Dia ⇒ $\overline{lh} + \overline{slh} + 2$

return height

→ save ⇒ largest diameter

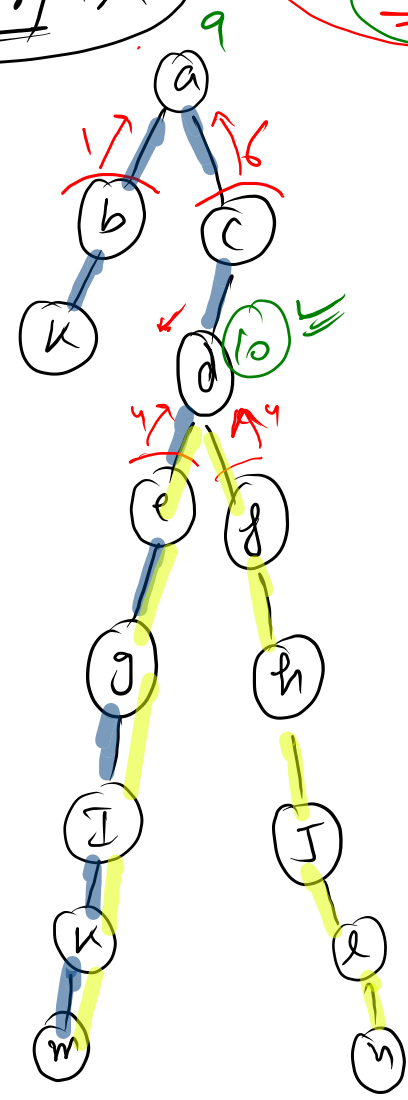
static

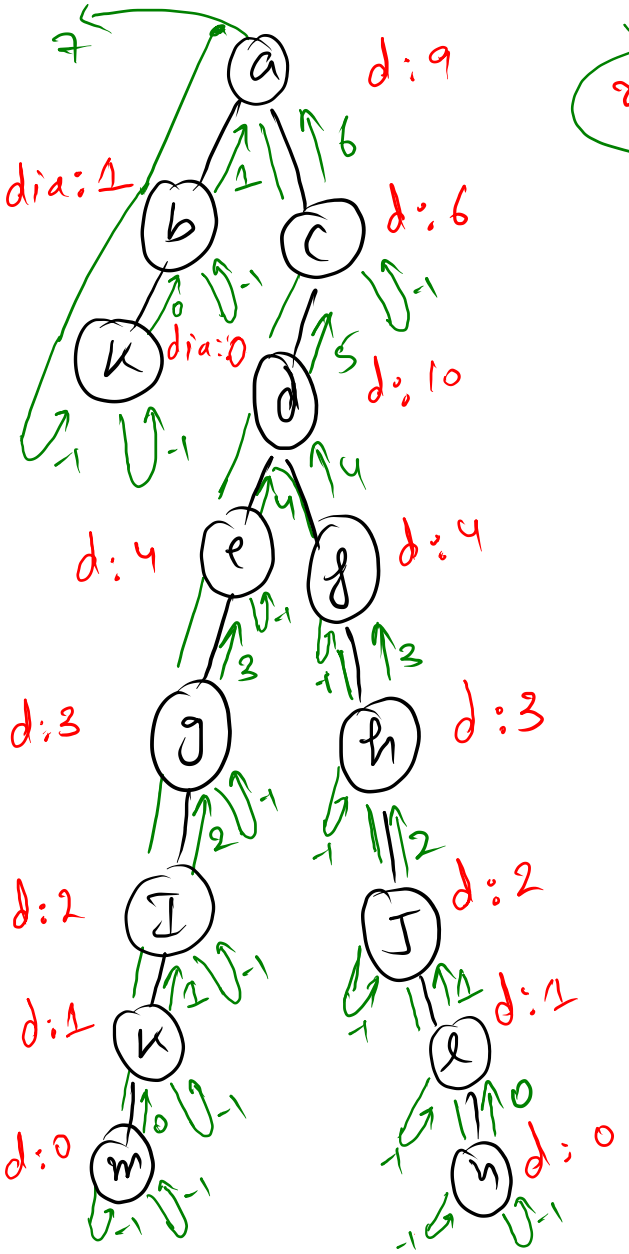


Binary Tree

DiaOfNode

→ $\overline{lH} + \overline{RH} + 2$





res = ∅ ~~1~~ ~~2~~ ~~3~~ ~~4~~ 10

```java
static int res;
public static int diameter1(Node node) { // return height
  if(node == null){
      return -1;
  }

  int lh = diameter1(node.left);
  int rh = diameter1(node.right);

  int dia = lh + rh + 2;
  if(dia > res){
      res = dia;
  }

  return Math.max(lh,rh)+1;
}
```

```java
res = 0;
diameter1(root);
System.out.println(res);
```

```
19
50 25 12 n n 37 30 n n n 75 62 n 70 n n 87 n n
```

0+55+0+3 0+70+45+0+190

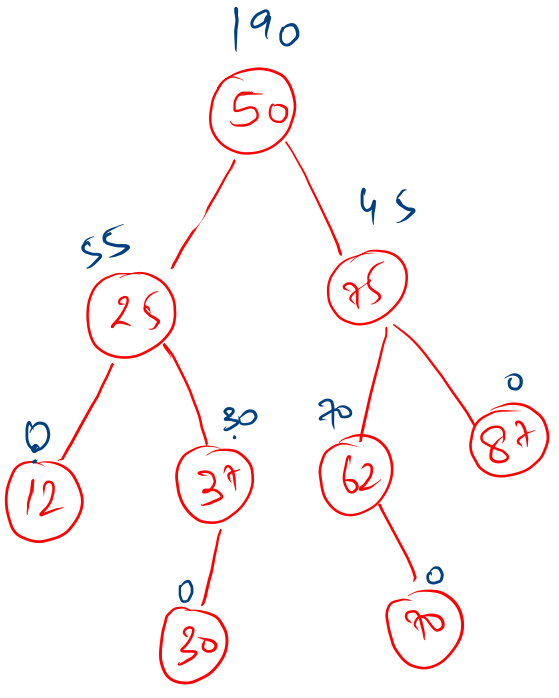
⇒ 390

```
static int tilt = 0;
public static int tilt(Node node){
  // write your code here to set the tilt data member
}

public static void main(String[] args) throws Exception {
  BufferedReader br = new BufferedReader(new InputStreamReader(System.in))
  int n = Integer.parseInt(br.readLine());
  Integer[] arr = new Integer[n];
  String[] values = br.readLine().split(" ");
  for (int i = 0; i < n; i++) {
    if (values[i].equals("n") == false) {
      arr[i] = Integer.parseInt(values[i]);
    } else {
      arr[i] = null;
    }
  }

  Node root = construct(arr);

  tilt(root);
  System.out.println(tilt);
}
```



Tilt of a node

Sum(left) - Sum(right)

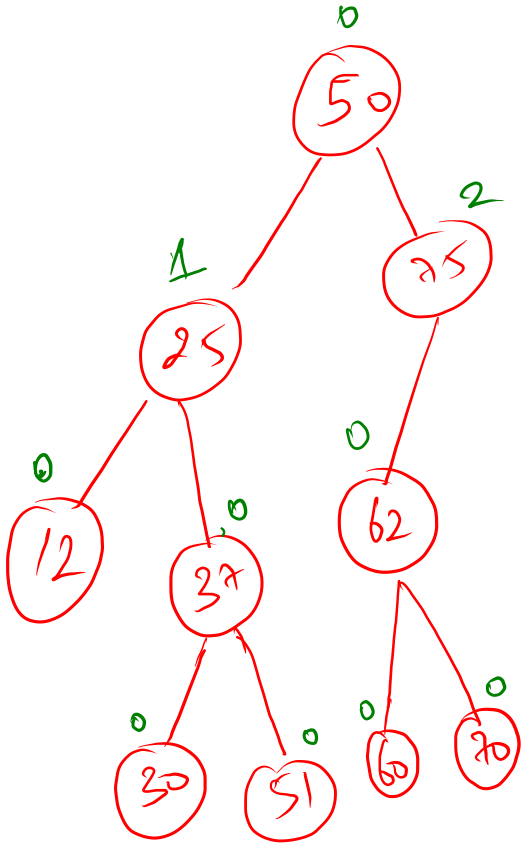
false

height

BF

Diameter

Tilt



res = true false

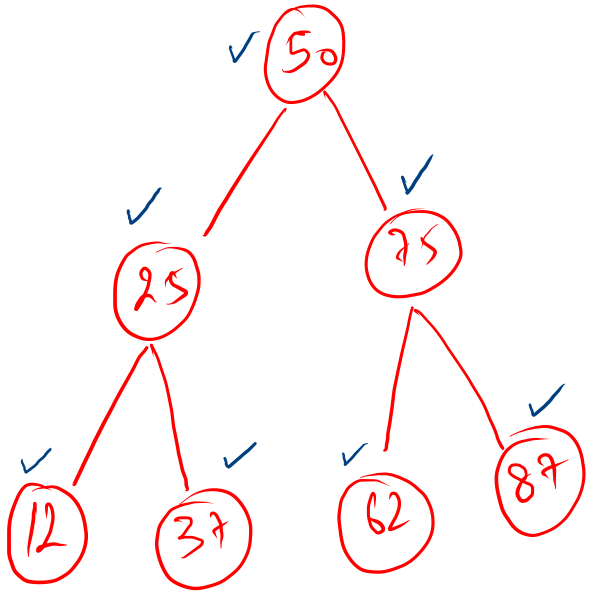
Balance

BF = | Height(left) − Height(right) |

BF ≤ 1 ∀ nodes

true (balanced)

false (unbalanced)



BST Property

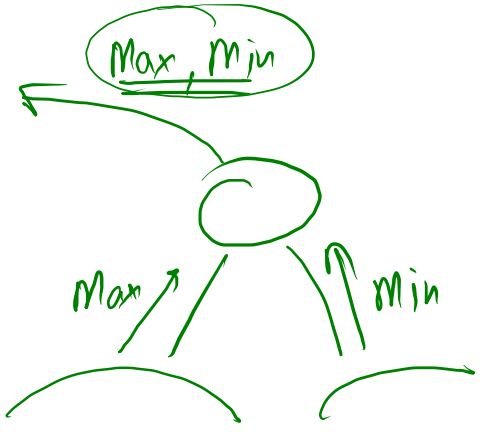
left < node.data < right

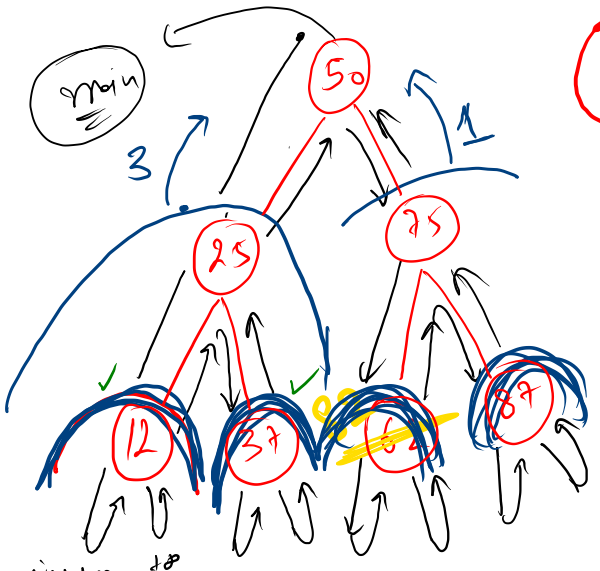
BST Property + * nodes

⇓

BST

Max (Left) < node.data < Min (Right)





isBST

BST Prop

left.max < n.data < Right.min

Left. isbst

right. isbst

l BST → 3 25

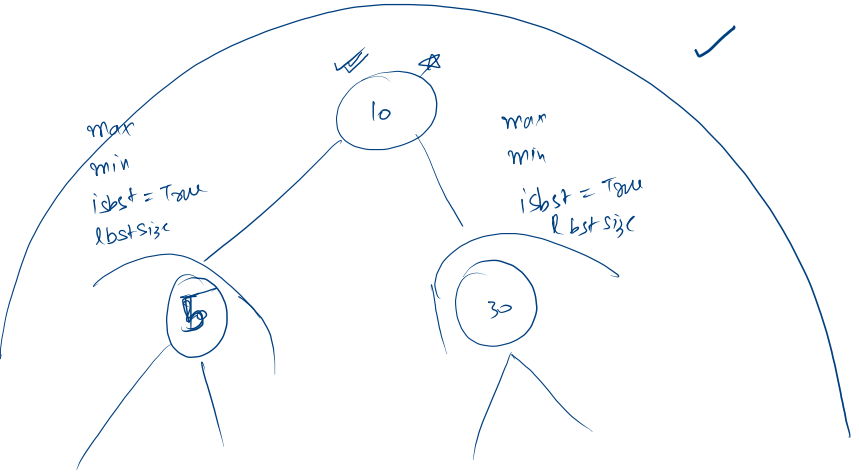
H.W.

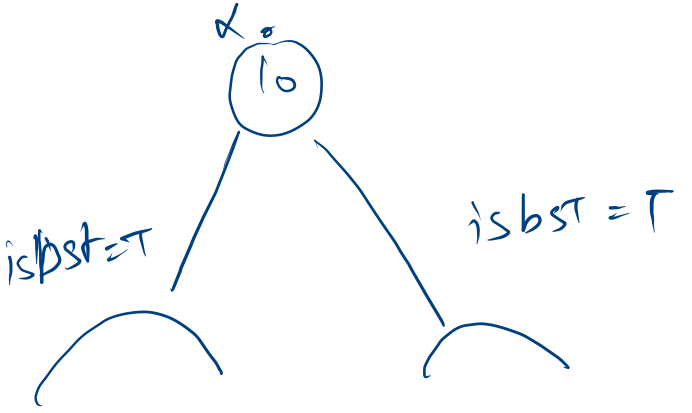
null
12
37
l 25
~~82~~ ~~62~~
87
75
⇒ 50

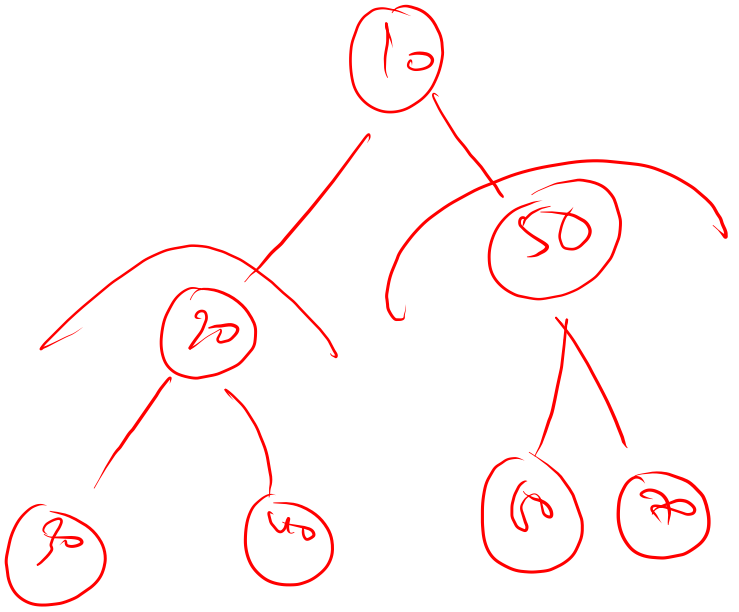
isbst?

Max	Min	Is Bst	lbstsize	lbstNode
-∞	+∞	True	0	null
12	12	True	1	12
37	37	True	1	37
37	12	True	3	25
~~82~~ ~~62~~	~~82~~ ~~62~~	True	1	82
87	87	True	1	87
87	62	~~True~~ false	1	82
87	12	~~True~~	3	25

false


10
max
min
isbst = True
lbstSize
5
max
min
isbst = True
lbstSize
30


10
isbst=T
isbst=T



Size => ?