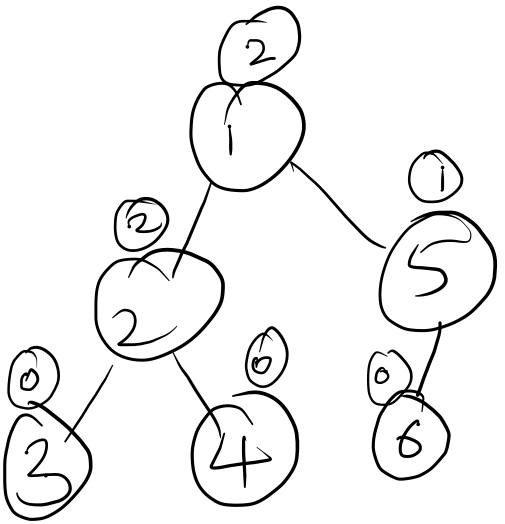
BINARY TREE

→ Atmost 2 children

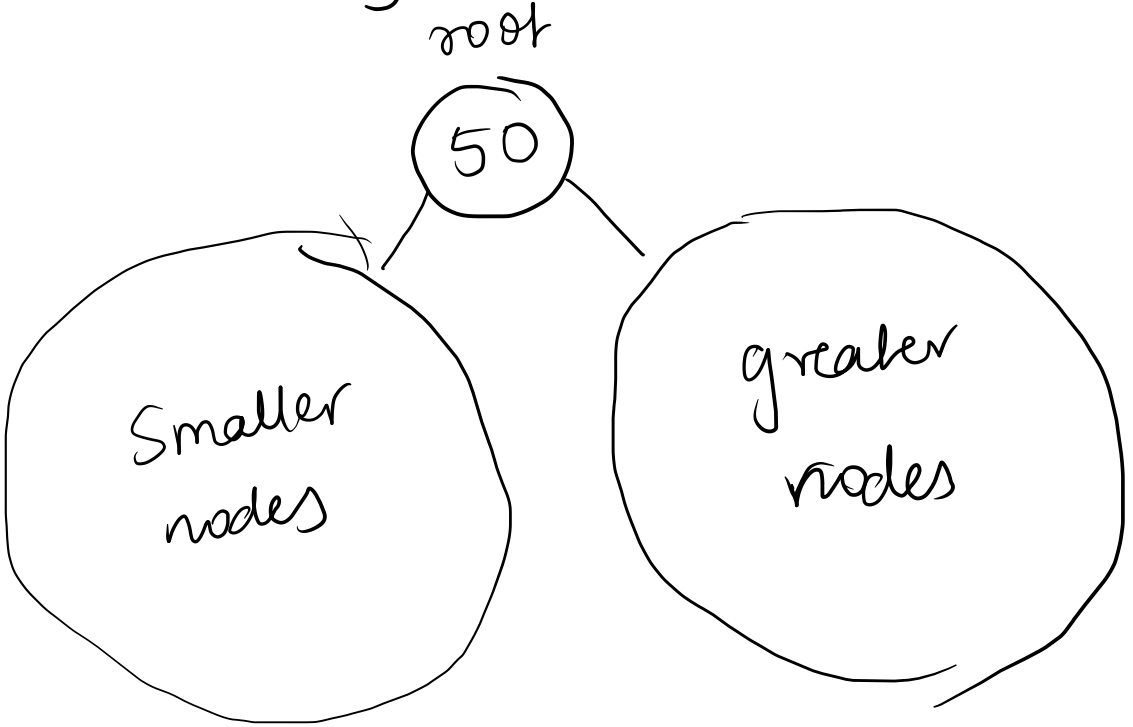
(0, 1, 2)

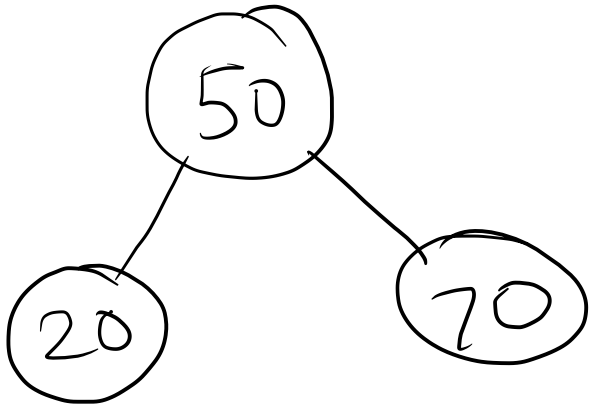
child nodes



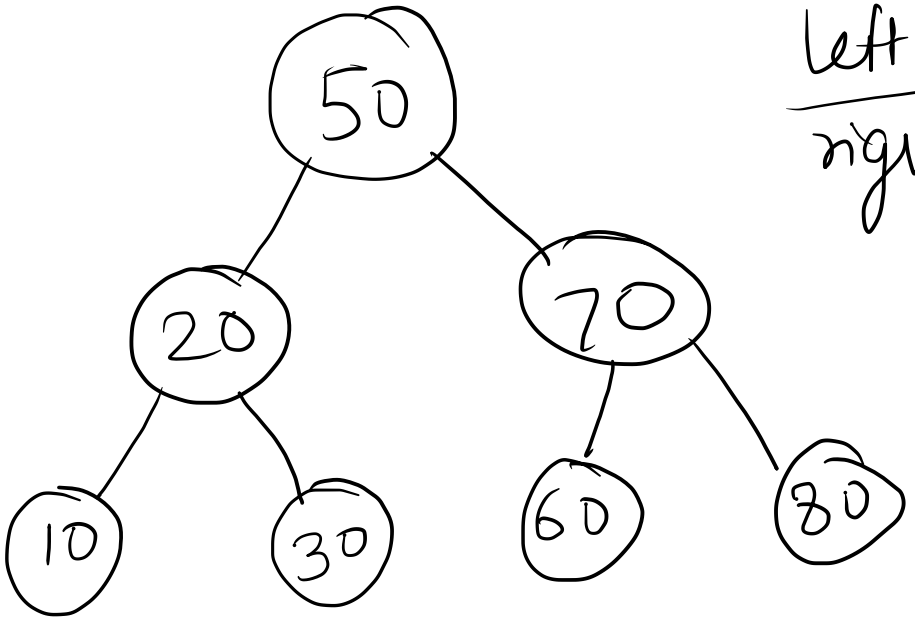
BINARY SEARCH TREE

is a ~~Binary tree~~





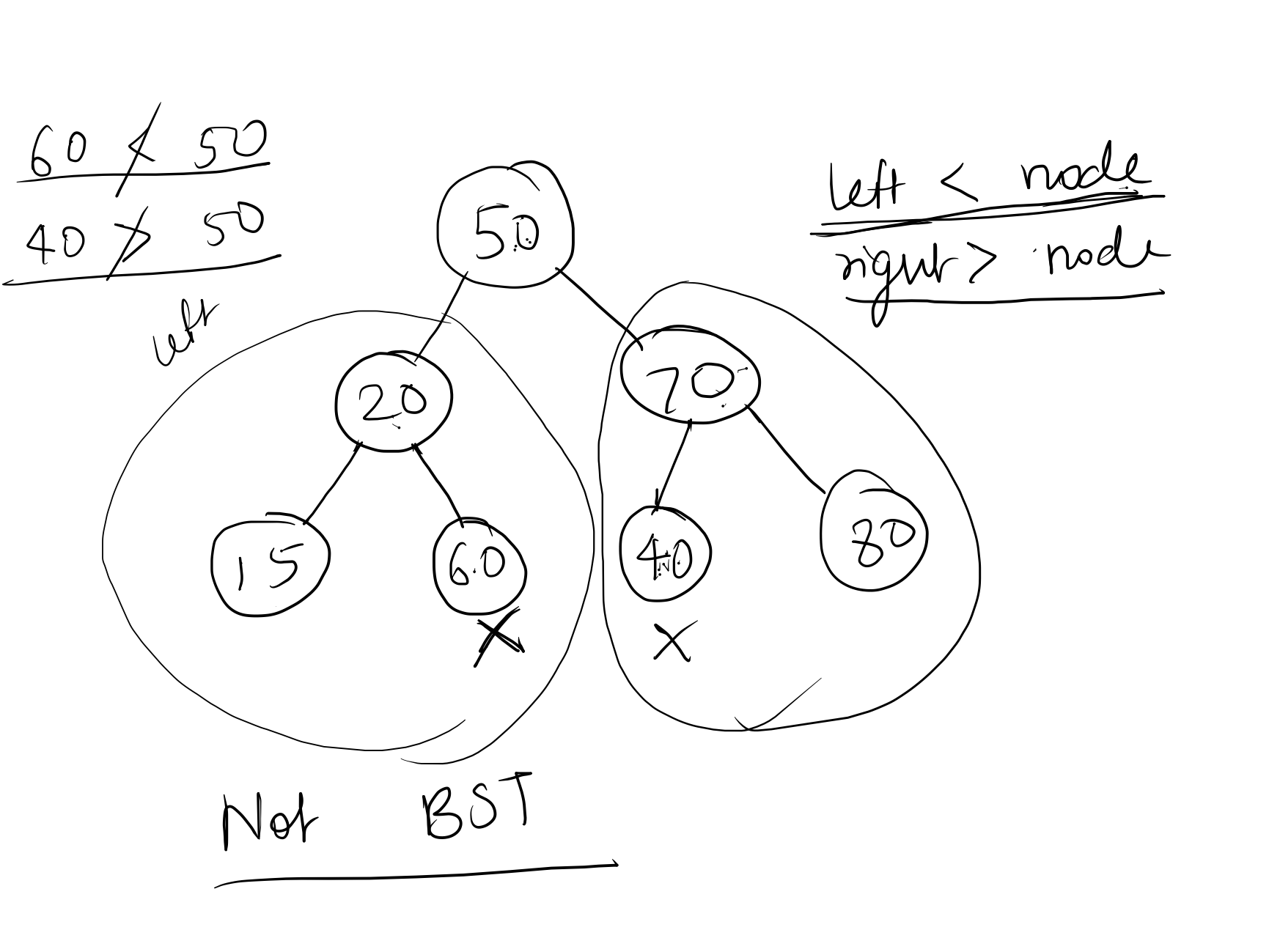
Binary Search Tree

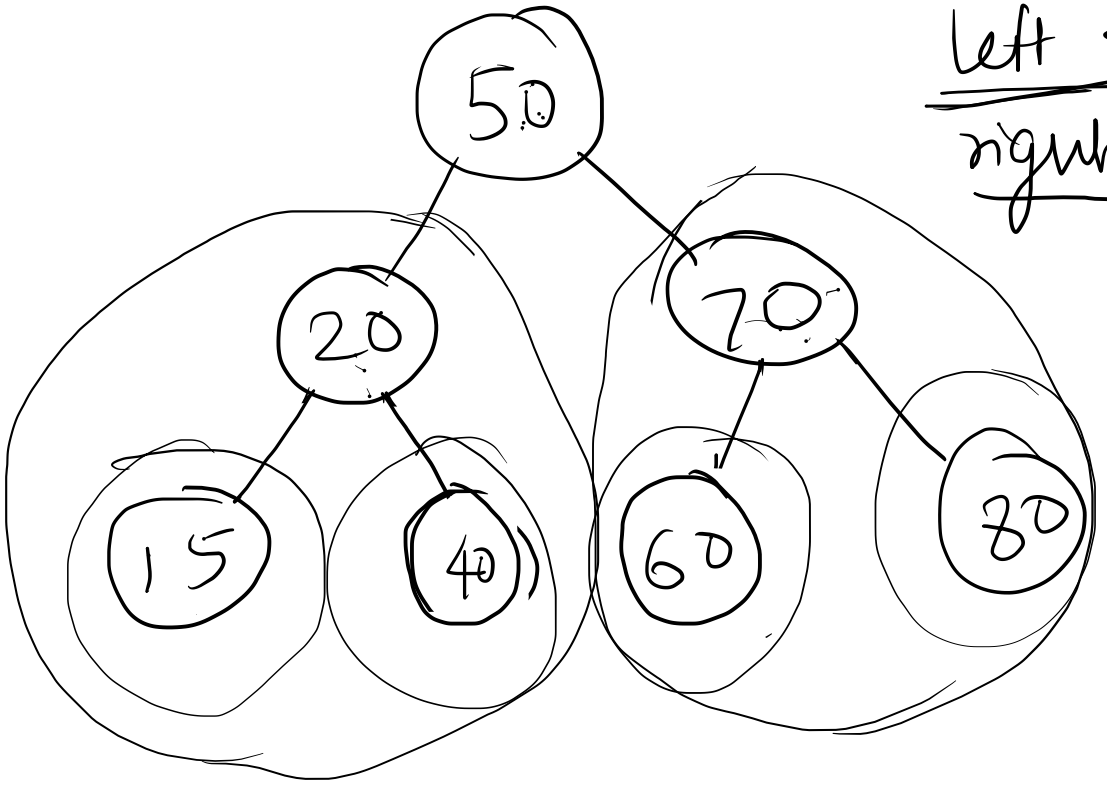


left < node

right > node

Binary Search Tree

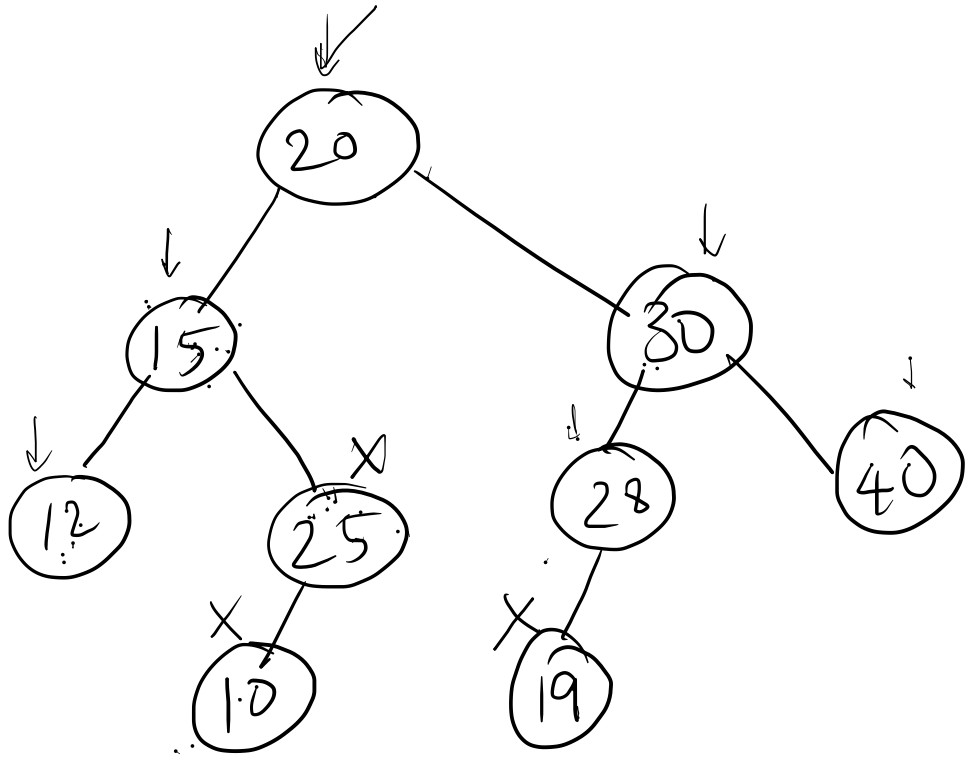

60 ≯ 50
40 ≯ 50
left < node
right > node
left
50
20
70
15
60
40
80
Not BST



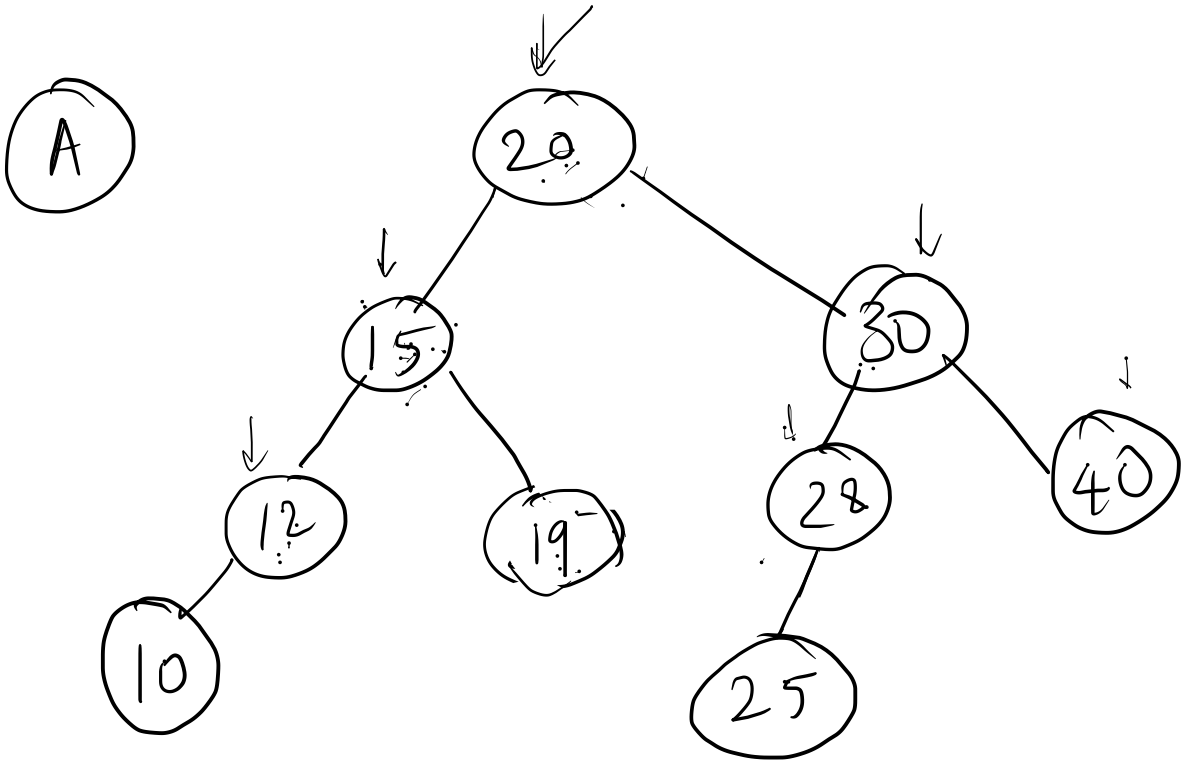
left < node

right > node

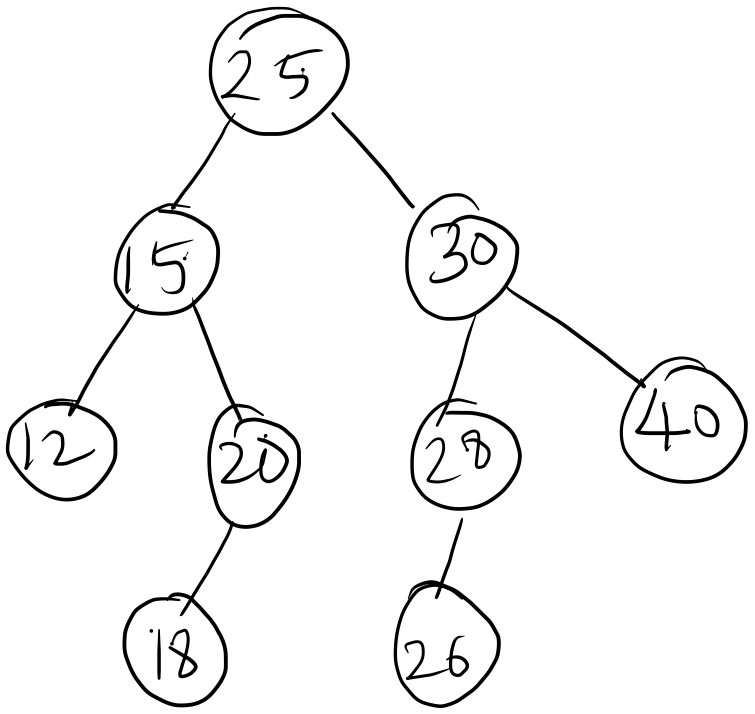
A



NOT BST


A
20
15
30
12
19
28
40
10
25

B



left < node

right > node

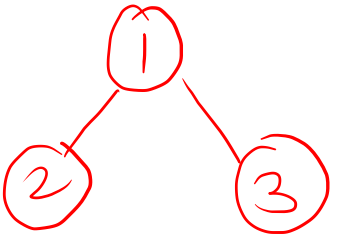
Yes, it is BST

A.) Every BST is a Binary Tree

TRUE

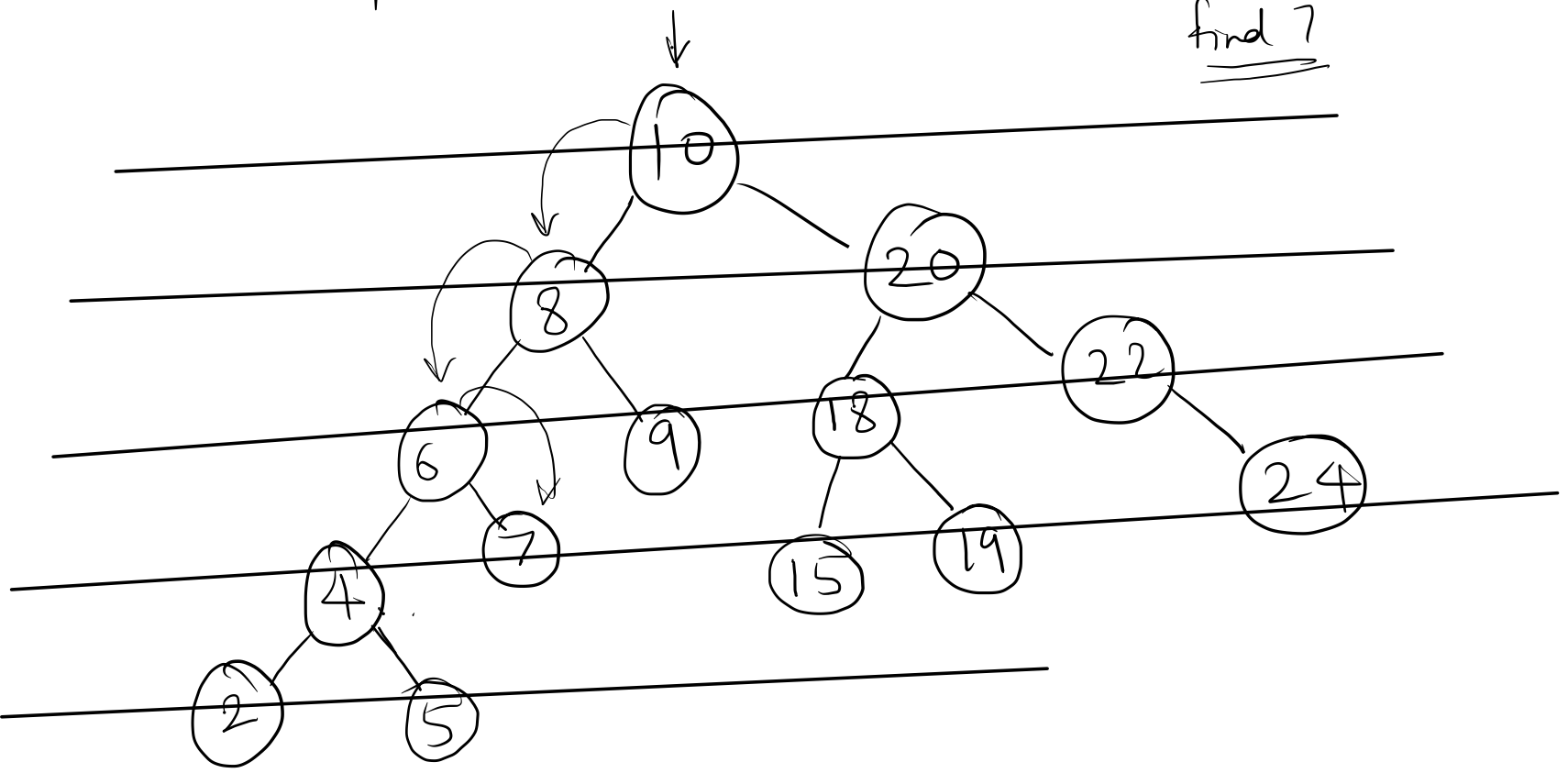
B.) Every Binary Tree is a BST

FALSE

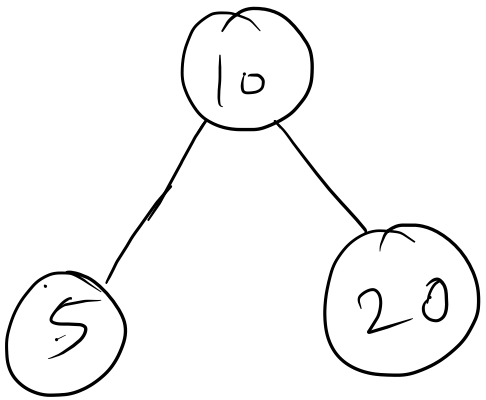


Why called "search" tree?

find 7



Inorder → sorted



Preorder → Root, left, right
10, 5, 20

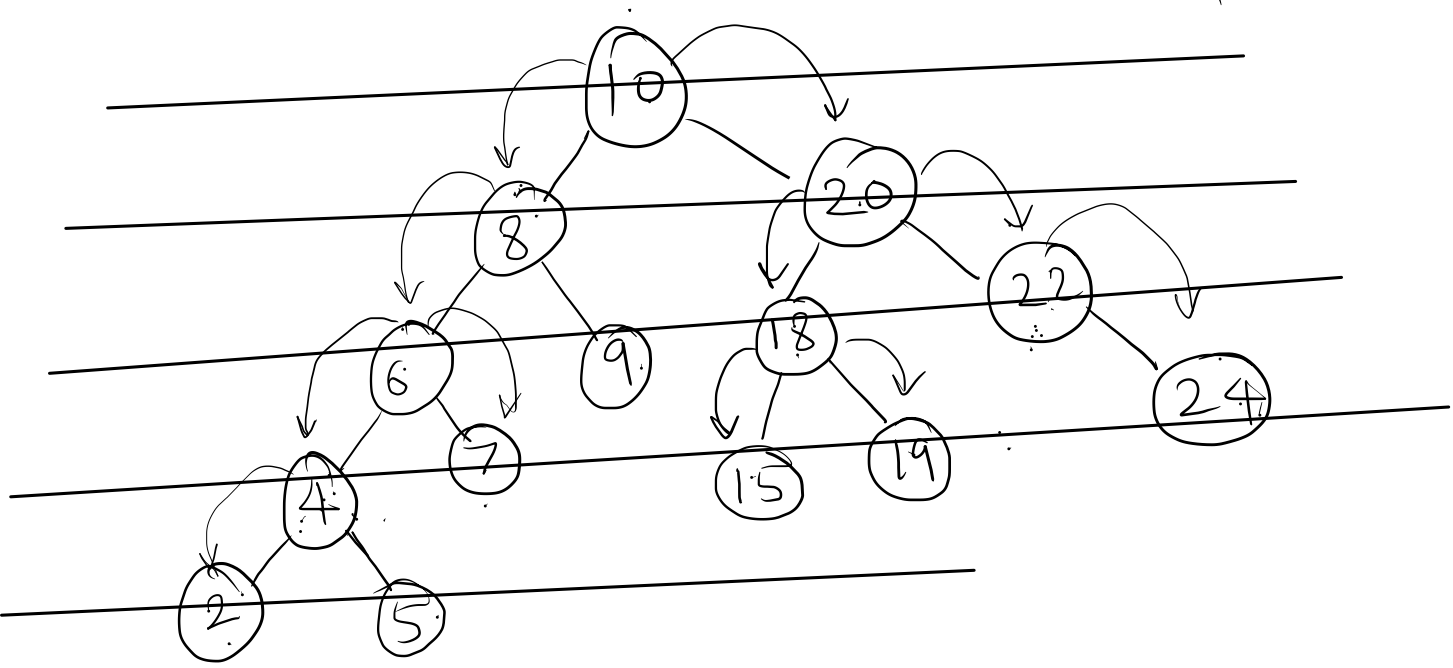
Inorder → Left, Root, right
5, 10, 20

Postorder → left, right, root
5, 20, 10

Inorder → left, root, right

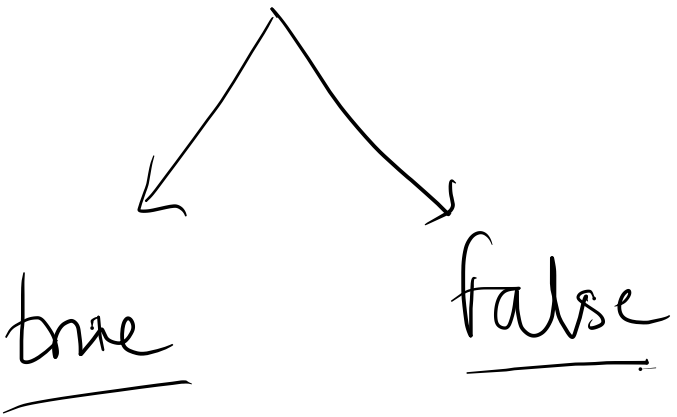
2, 4, 5, 6, 7, 8, 9, 10, 15, 18, 19, 20, 22, 24

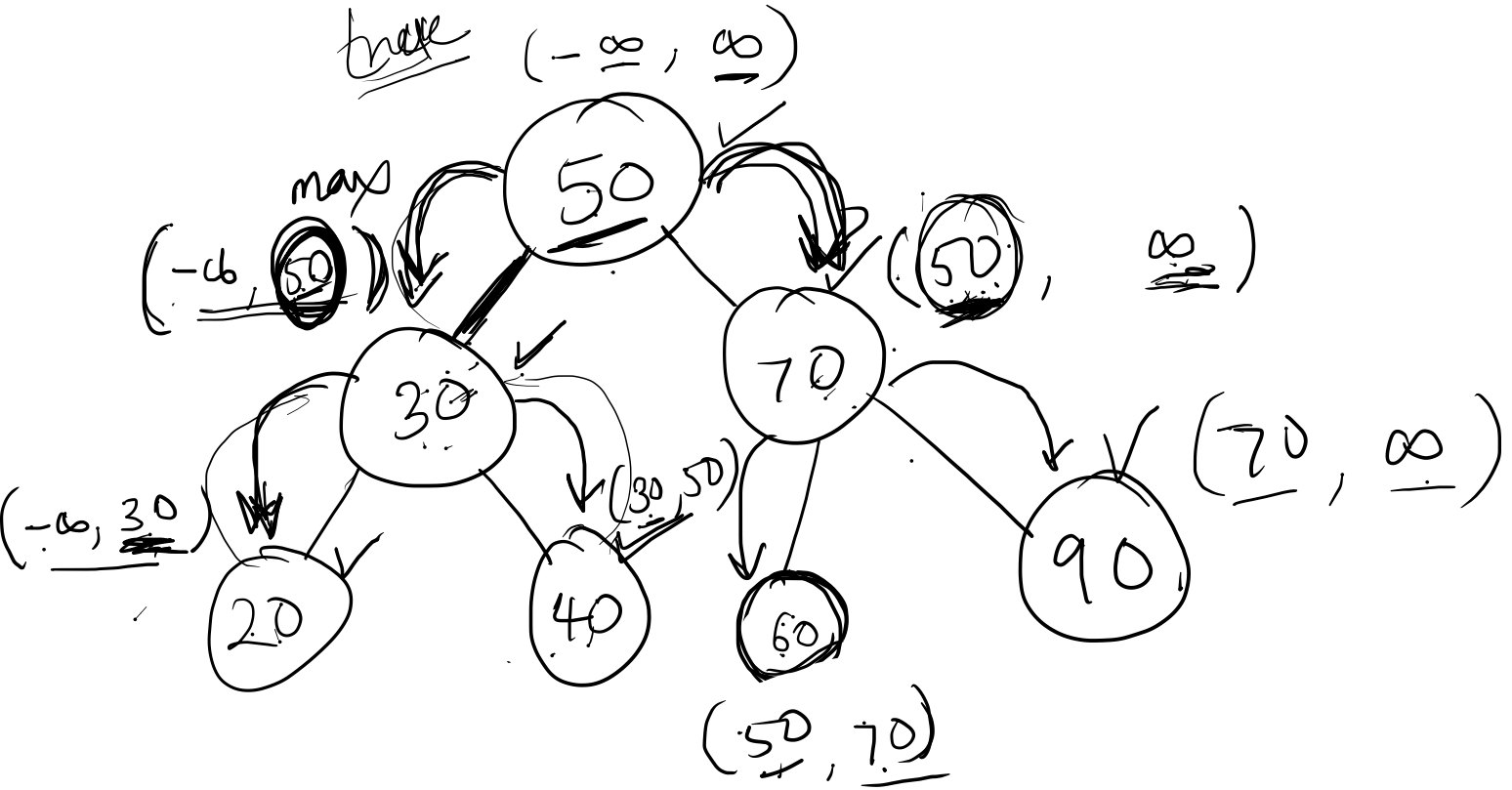
sorted

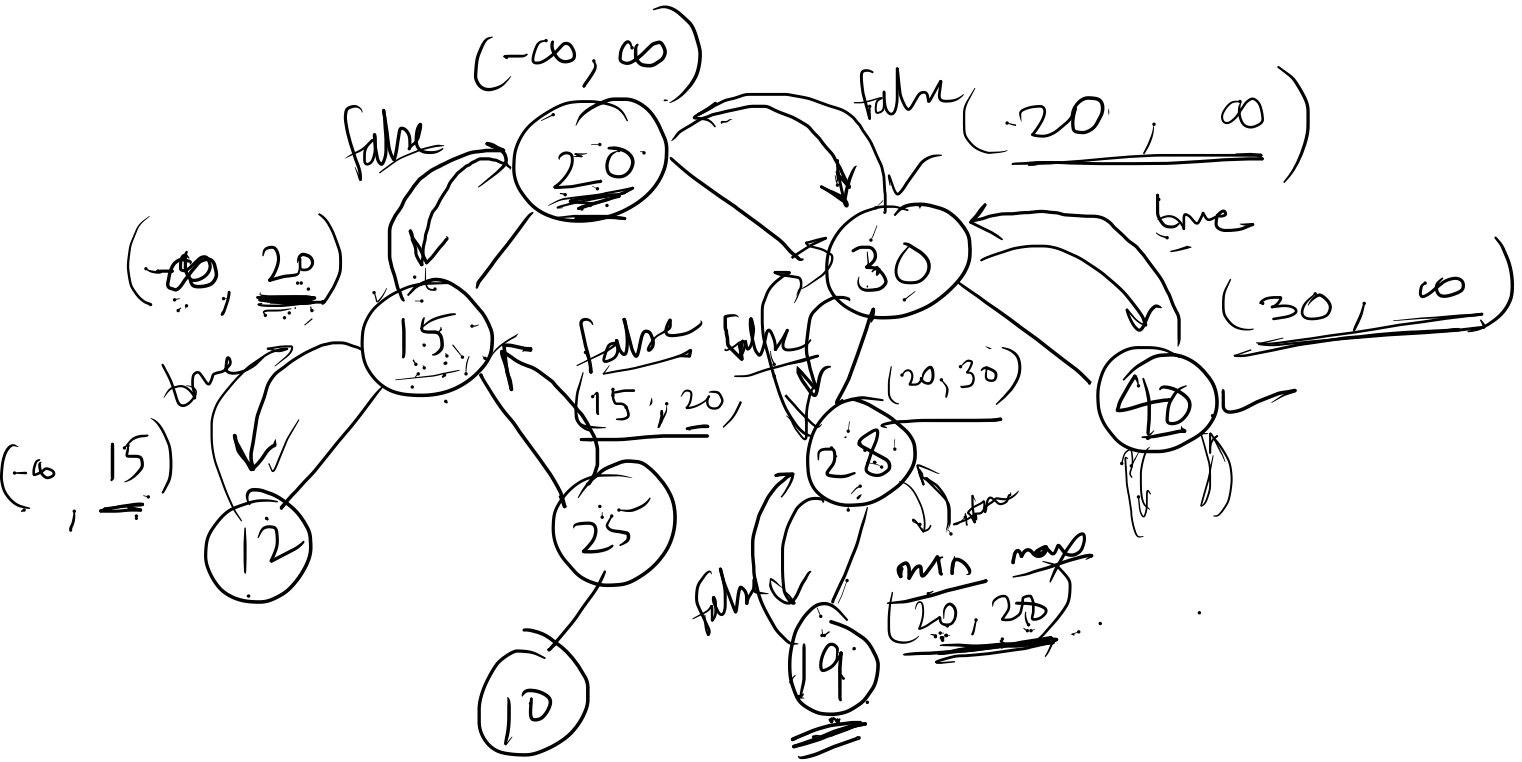


* Inorder Traversal of a BST gives us the sorted order

Validate if tree is BST ?

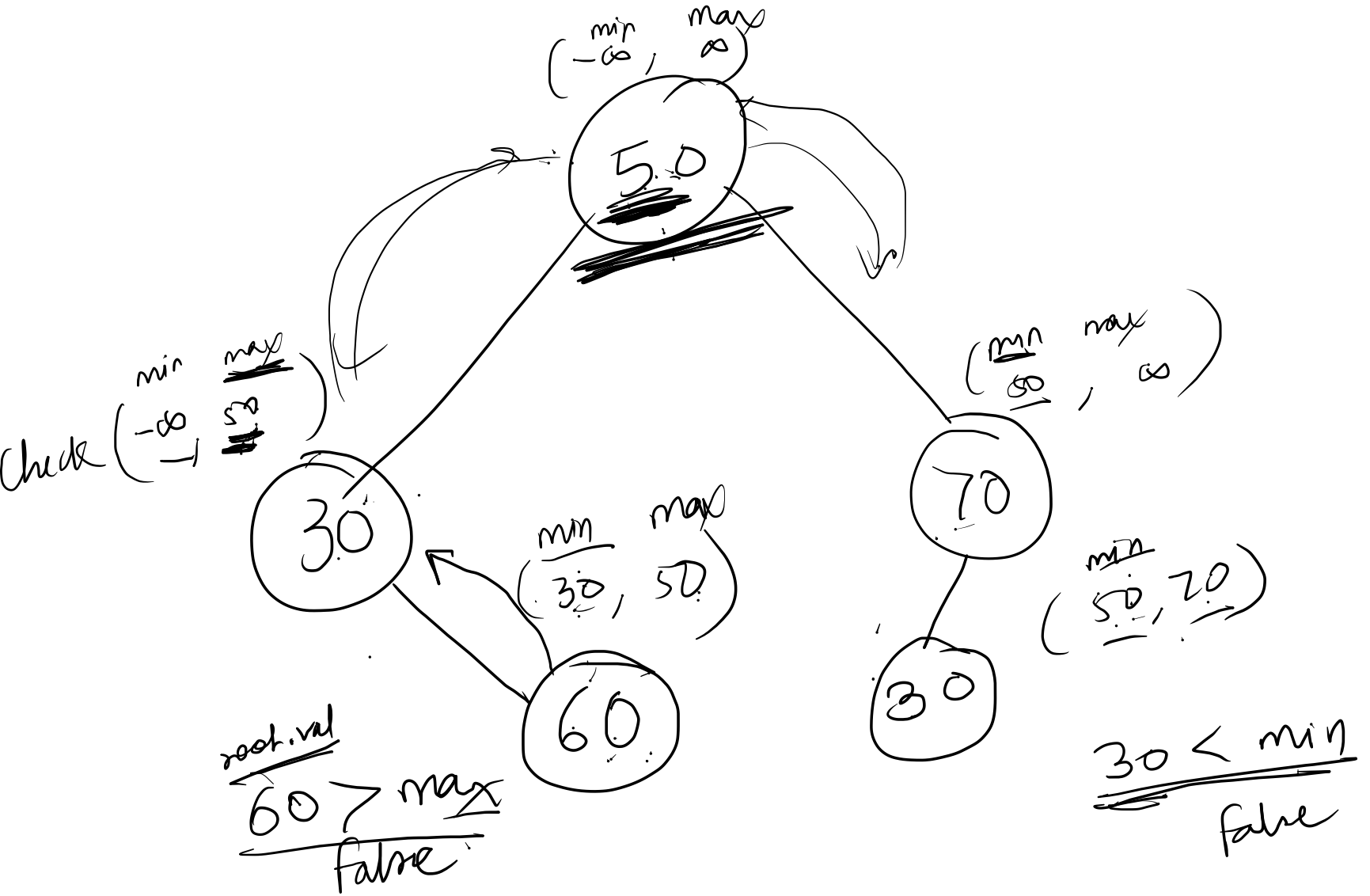



trace
(-∞, ∞)
50
max
(-∞, 50)
(50, ∞)
30
70
(-∞, 30)
(30, 50)
(70, ∞)
20
40
60
90
(50, 70)


(-∞, ∞)
20
false
false
(20, ∞)
(-∞, 20)
15
30
true
(30, ∞)
false
false
(15, 20)
(20, 30)
40
28
(-∞, 15)
12
25
min
max
20, 25
19
10

ans = false

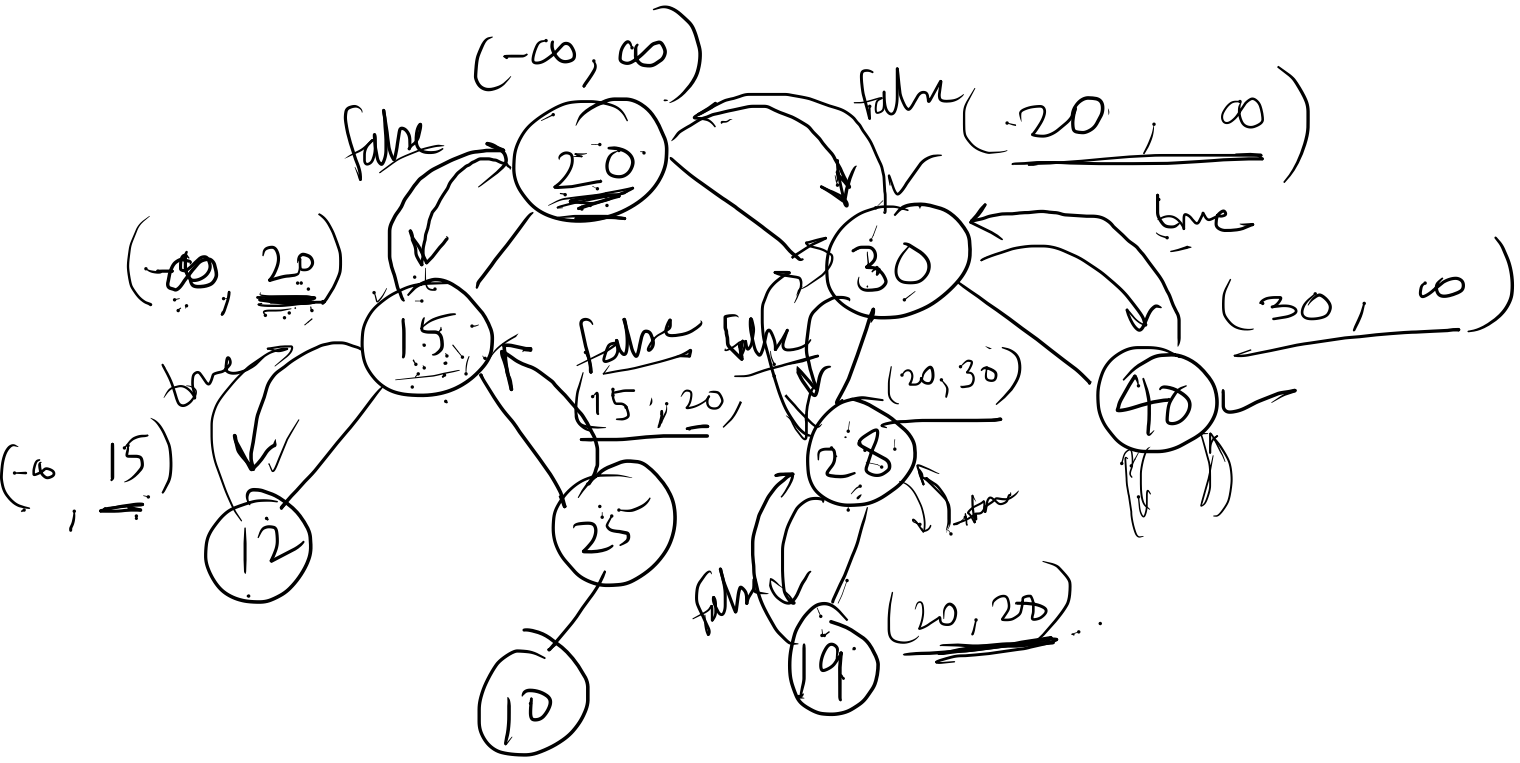
```
if (root == null)
    return true
if (root.val <= min || root.val >= max)
    return false
```


min
max
(-∞, ∞)
50
min
max
(-∞, 50)
Check
30
min
max
(50, ∞)
70
min
max
(30, 50)
60
min
max
(50, 70)
30
root.val
60 > max
false
30 < min
false

T.C = O(n)

Aux SC = O(n)

recursive stack


(-∞, ∞)
20
false
false
(20, ∞)
(-∞, 20)
15
30
true
true
(30, ∞)
false
false
(15, 20)
(20, 30)
40
28
(-∞, 15)
12
25
false
(20, 28)
19
10

ans = false