Binary Search Trees

DnC

1) Binary search

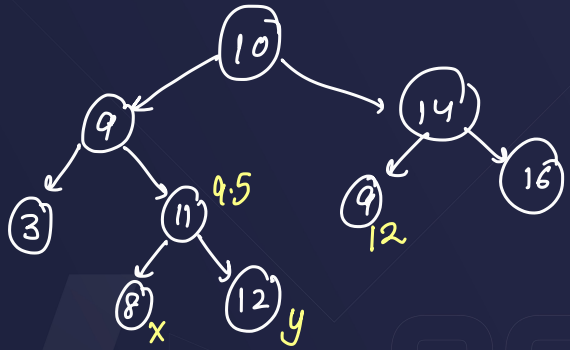
2) Binary Trees

any BT is BST if → root.val < every value in RST

~~root~~.val > every val in LST

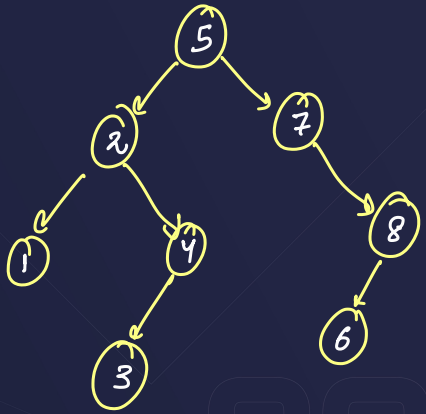
If it is BST or not.

Rules of BST



$9 < x < 9.5$

$9.5 < y < 10$

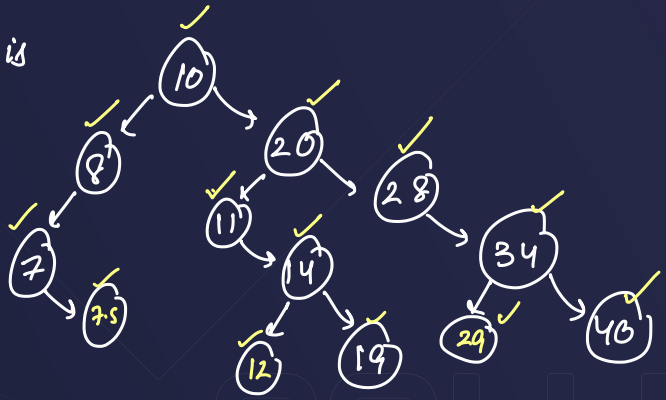


you have to fill the tree such that

1, 2, 3, 4 | 6, 7, 8

Is this a BST.

If not what is wrong



In a BST all values are unique → always

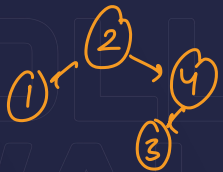
Q#1 I have to make BST's of 2 nodes/size & values being 1 & 2. How many diff. BST's can exist?

Q#2 What about 3 nodes → 1, 2, 3. How many BST's

2 nodes - 1 & 2 . How many Binary Trees ?

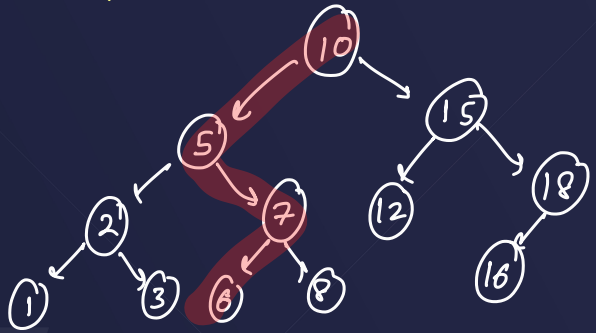
4 trees

Homework : With 4



Q/ Search in a B.S.T.

(root, target)



tar = 4

8

Balanced → O(log n)

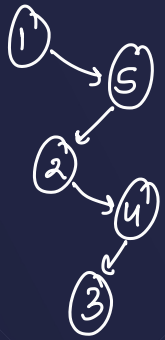
target = 1

↓

T.C. = O(n)

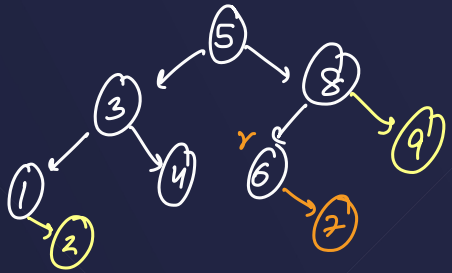
T.C. = O(h)

for every case



Degenerate Tree

O(n)



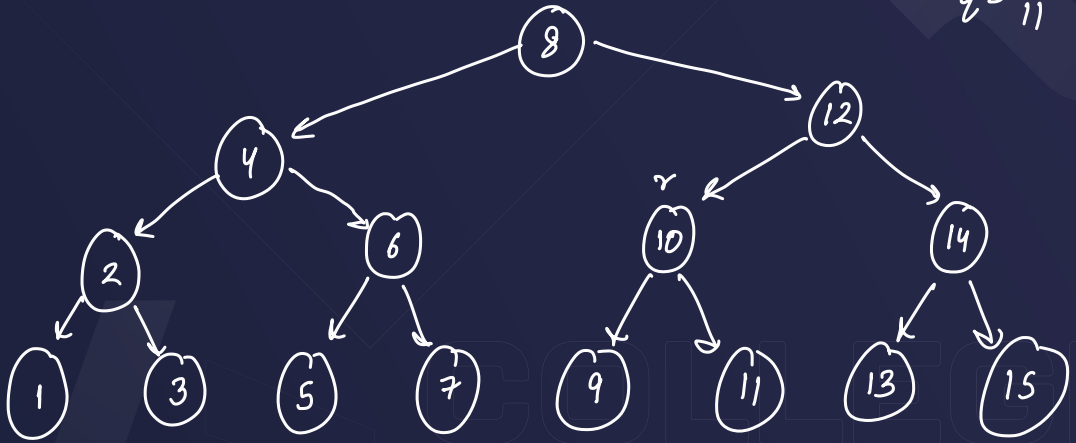
if we have to insert 7

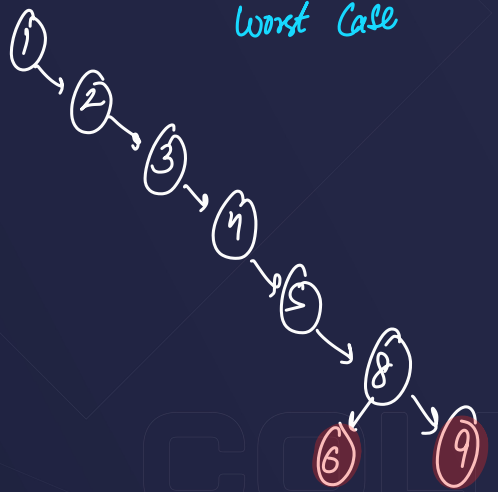
val = 9
val = 2
val = 7

Inorder traversal of a BST is always sorted

LCA of a BST

P = 9
Q = 11




Worst Case
1
2
3
4
5
8
6
9

Validate a BST (is BST)

T.C. = O(n)

S.C. = O(h)

In : N 1 2 4 5 6 8 9

prev

Node prev = NULL;

```java
public void inorder(TreeNode root){
    if(root==null) return;
    inorder(root.left);
    if(prev!=null){
        if(root.val<=prev.val){
            flag = false;
            return;
        }
    }
    prev = root;
    inorder(root.right);
}
```

N 1 2 3 5 4 6 7 8

p

flag = ~~true~~ false