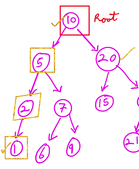
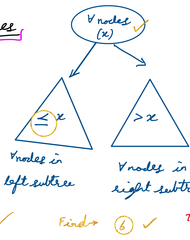
BST

Friday, 24 September 2021 9:05 PM

Binary Search Trees

BST is useful for search operations.





Find → 6 ✓ TC = O(H)

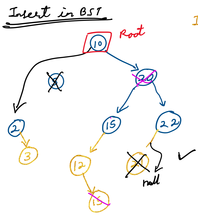
5 ✓

8 ✗

Find → smallest value = 1 TC = O(H)

largest value = 2 ✓

Insert in BST



Insert → 12 ✓ TC = O(H)

21 ✓ SC = O(1)

15 ✓ 3 ✓

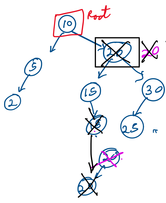
Q → how many nodes have value = 15.

Ans = 2

Deletion in BST → 1) delete leaf Eg → 21 } TC = O(H)

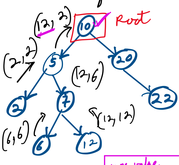
2) delete node with single child Eg → 5

3) delete node with 2 children. Eg → 20 TC = O(H) ✓



TC = O(count * H)

Q → check if the given BT is a BST? Amazon | Microsoft



left child ≤ x < right child ✗

✓ Inorder traversal is sorted ✗

2 5 6 7 10 10 10 20 22

max value in left subtree ≤ ∀ nodes ≤ min value in right subtree ✓

TC = O(N)

SC = O(H)

```
isBST = true;   if = 2 ?;
pair max_min (root) {
    if (root == null)  return {-inf, inf};
→   pair l = max_min (root.left);
    pair r = max_min (root.right);
x→  if( l.max > root.data || root.data >= r.min)
        isBST = false;
→   return {max (root.data, r.max),
            min (root.data, l.min)};
}
```

pair { max; min; } -inf inf -inf inf

7 12 7 6

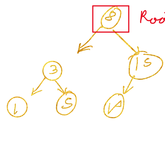
Q → construct balanced BST from sorted array?

$|h_{left} - h_{right}| \le 1$

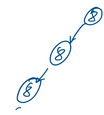
A = [1, 3, 5, 8, 10, 15]

TC = O(N)

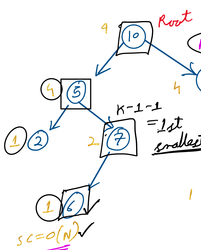
SC = O(1)



A = [8 8 8 8 8 8 8 8] X



Q → Find K^{th} smallest element in BST?



Ans = K^{th} element of inorder traversal.

TC = O(N)

K = 3 Ans = 6

Find K^{th} smallest for multiple queries?

Inorder Traversal

→ Every time TC = O(N * Q)

→ Store TC = O(N + Q) ✓

SC = O(N)

SC = O(N) ✓

K = 7

```
if(# in left subtree == K-1)
        current node is ans.
if (# in left subtree >= K)
        go left;
else {
        go right;
        K = K - # in left subtree - 1;
}
```

TC = O(H * Q)

I current node