Trees → Pointers

↓

→ Binary Search Trees (BST)
→ AVL (Balanced BSTs)
→ Further Reading.

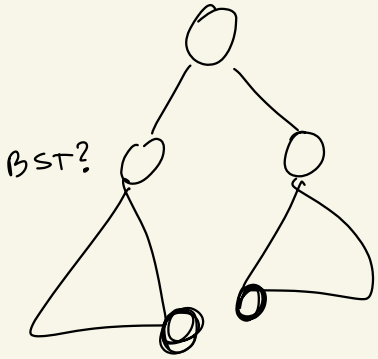
Why BST?

→ search
→ Insert
→ Delete.
→ Traverse Sorted.
→ Closest.
→ Rank

	Search	Insert	Deletes	Closest Value.	Sorted Traversal
Array (unsorted)	$O(n)$	$O(1)$	$O(n)$	$O(n)$	$O(n \log n)$
Array (Sorted)	$O(\log n)$	$O(n)$	$O(n)$	$O(\log n)$	$O(n)$
Linked List	$O(n)$	$O(1)$ / $O(n)$	$O(1)$ / $O(n)$	$O(n)$	$O(n \log n)$
Hash Table	$O(1)$	$O(1)$	$O(1)$	$O(n)$	$O(n \log n)$
BST	$O(\log n)$	$O(\log n)$	$O(\log n)$	$O(\log n)$	$O(n)$

→ keys on the left subtree < root
key on the right subtree > root
} recursively for all nodes.

rightmost node of the left.



When in an interview / coding round → Dont code yourself.

C++ → sets, maps, ordered sets, multimaps.

How are these implemented?

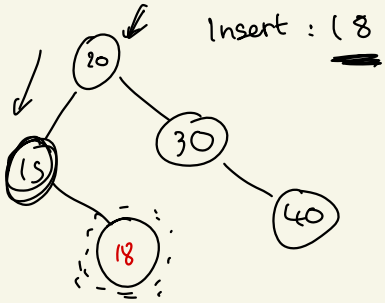
Java → Treeset, Treemap.

Balanced BSTs (Red-Black) } → AVL.

Search: Iteratively - O(1)

O(h) ← stack space. Recursive.

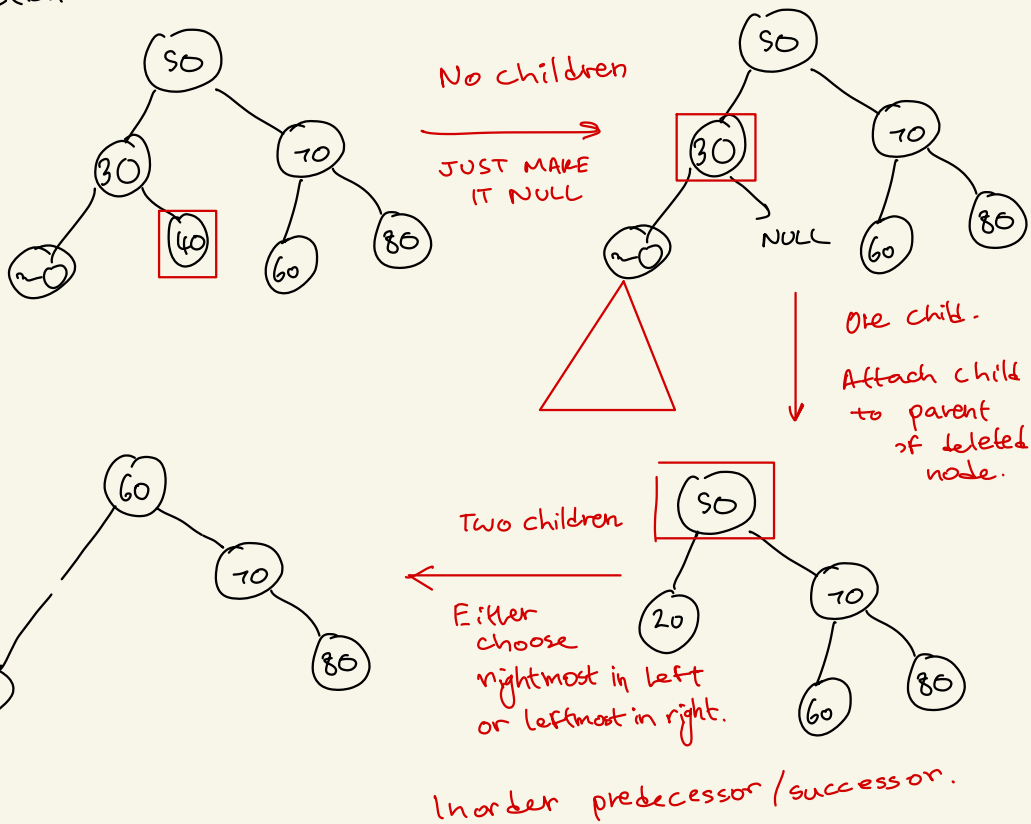
Insert: 20, 15, 30, 40




```
bool search (Node* root, int x){
    Node* cur = root;
    while ( cur != NULL){
        if ( cur -> key == x)
            return true;
        if ( cur -> key > x)
            cur = cur -> left;
        else
            cur = cur -> right;
    }
    return false;
```

↳ HW: Change for insert.

Deletion:



Inorder predecessor / successor.

HW: Implement these iteratively

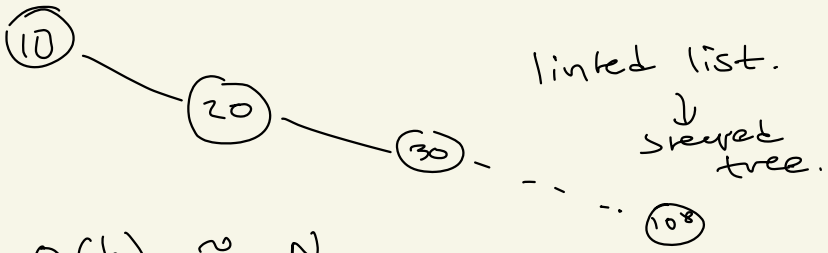
Q: Given x output the greatest element in the tree that is $\leq x$.

```
Node* floor ( Node* root, int x) {
    Node* ans = NULL;
    Node* cur = root;
    while (cur != NULL) {
        if ( cur -> key == x) {
            ans = cur;
            break;
        }
        if (cur -> key > x) {
            cur = cur -> left;
        }
        else {
            ans = cur;
            cur = cur -> right;
        }
    }
    return ans;
}
```

HW: Implement ceil(x).

Balanced BST

$10, 20, 30, 40 \cdots 10^8$

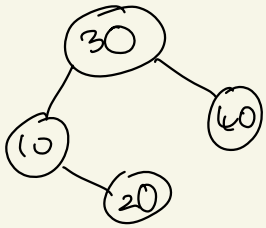


linked list.
↓
skewed tree.

$O(h) \approx N$

→ 30, 10, 40, 20

We don't know what order the keys arrive.



↓

Balanced BSTs

↓

Self-Balancing BSTs

AVL Tree

For each node:

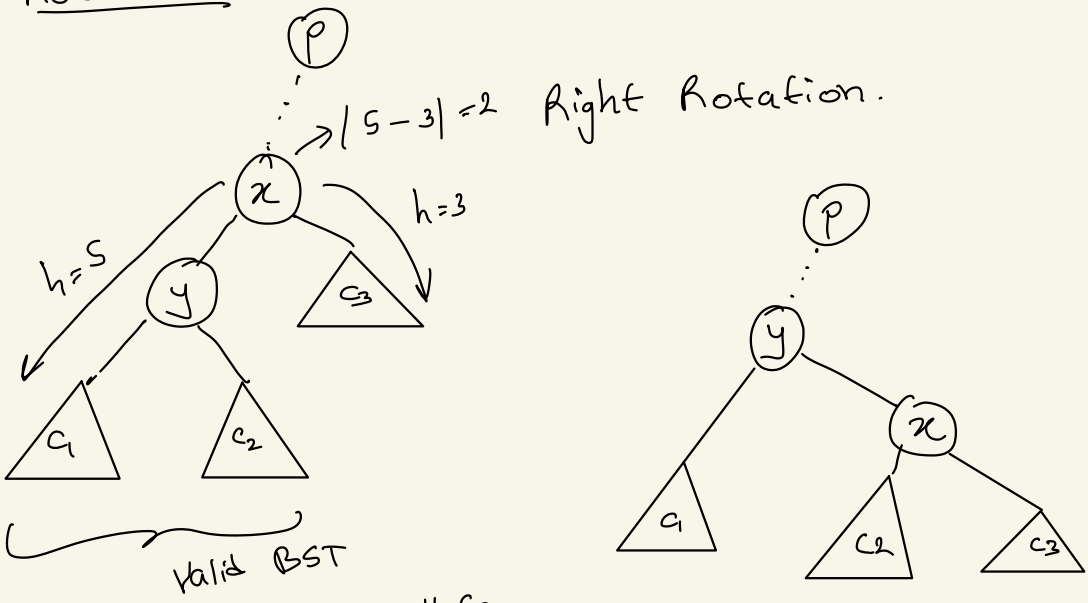
lh = height of left subtree

rh = " right "

AVL Property –

$|lh - rh| \leq 1$.

Rotations

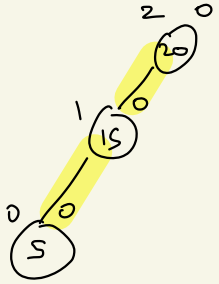


$y < x <$ all c_3

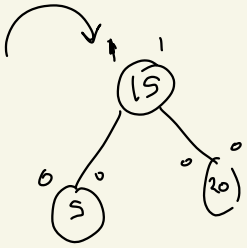
all $c_1 < x$

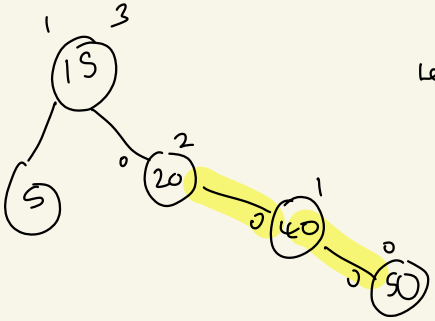
all $c_2 < x$ ✓✓

Insert: 20, 15, 5, 40, 50, 18

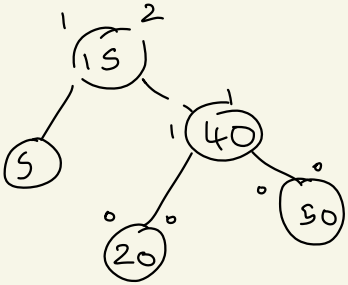


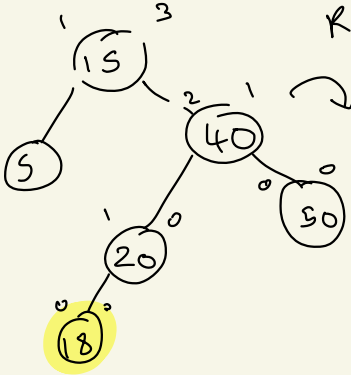
Right Rotation



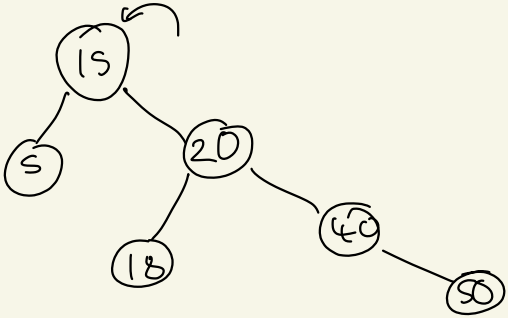


Left Rotation

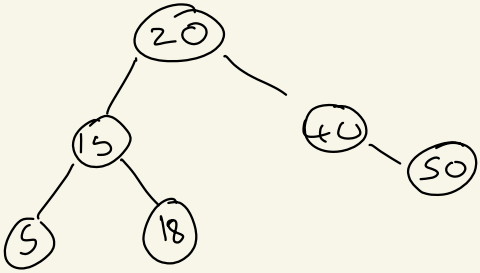




Right Rotate.



left rotate.



LL → Right Rotate
RR → Left Rotate.
} Single Rotation

LR → L, R
RL → R, L
} Double Rotation

Insert
Delete.
} Coding → HW

Further Reading

Assignment. → Problems on BSTs → ≥ 4-5
↳ Try solving without IDE.
↳ code iterative versions also.

Extra
↳ codingrounds. → AVL.
ordered_set. → policy based ds.
(know how to use, syntax).

optional Red - Black Trees.