Date
15th feb 2024
Thursday

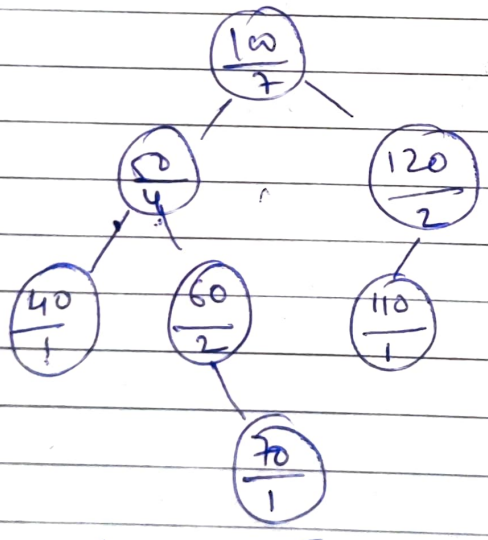
PAGE NO:
DATE:

Augmented Data Structure:

OS. select (x, i) → i^{th} smallest element in subtree rooted at x.

OS. Rank (T, x) → Size of left subtree + 1

Size[x] - No of nodes in subtree rooted at x.



O(n) → for finding i^{th} smallest element

↓ reduced to

O(log n)

r ← size of [left (x)] + 1

```
func OS-Select(x, i)
{   if i = r, then return x
    else if i < r, then return OS.Select(left[x], i)
    else           return OS.Select(right[x], i-r)
}
```

Complexity reduces to O(log n) for finding i^{th} smallest element

OS. Rank(T, x) :- finding rank of x, given pointer to node

```
r ← size of [left[x]] + 1
y ← x
while (y != root[T])
{
    if (y == right[p[y]])
    {
        r = r + size[left[p(y)]] + 1
    }
    y = p[y]
}
return r
```

OS. Rank(T, 50)

r ← 2

y = 50.

y = 100.

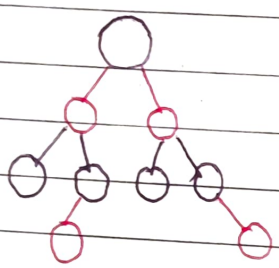
return 2

Date
20th feb 2024
Tuesday

PAGE NO.:
DATE:

Methodology for Augmenting DS's.

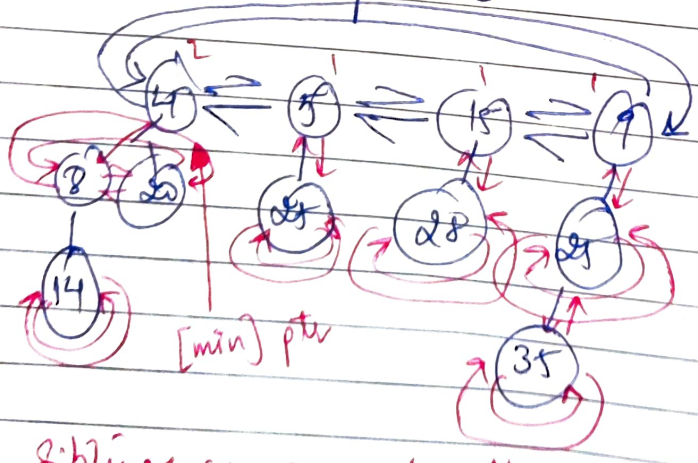
1. Choose the underlying DS.
2. Determine the additional info to maintain.
3. Verify that we can maintain the additional info on the DS - Efficient
4. Develop the new operations.



-: Fibonacchi Heap :-

collection of trees.
Each tree should follow heap ordering property

Root list is doubly linked



Siblings are connected through Double Circular LL.

degree : no. of child
mark of node : bool value (0/1)

Date
21st feb 2024
wednesday

4 + 14
5 + 14

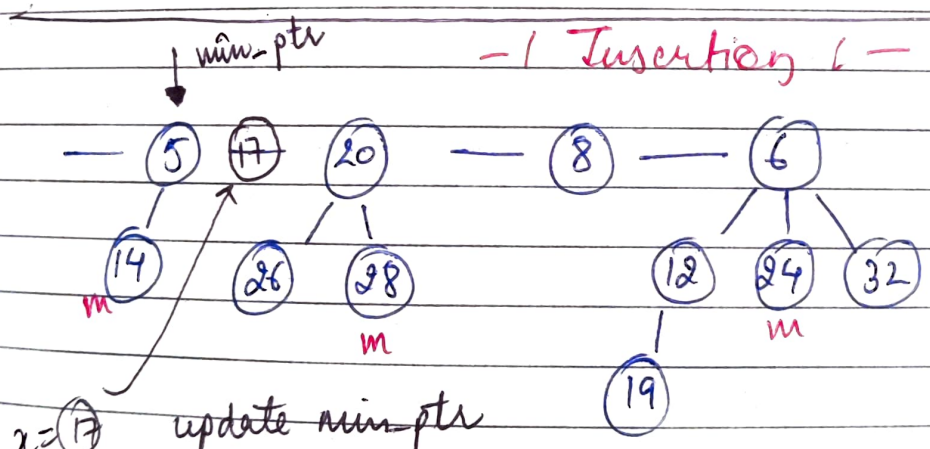
PAGE NO.:
DATE:

Fibonacci Heap:

Potential function, $\phi = t(H) + 2m(H)$
(t(H): no. of trees; m(H): no. of marked)

D(n) - upperbound on the degree of any node for a n node Fibonacci Heap

$D(n) \leq \log n$
$D(n) = O(\log n)$



— Insertion —

Insertion O(1)

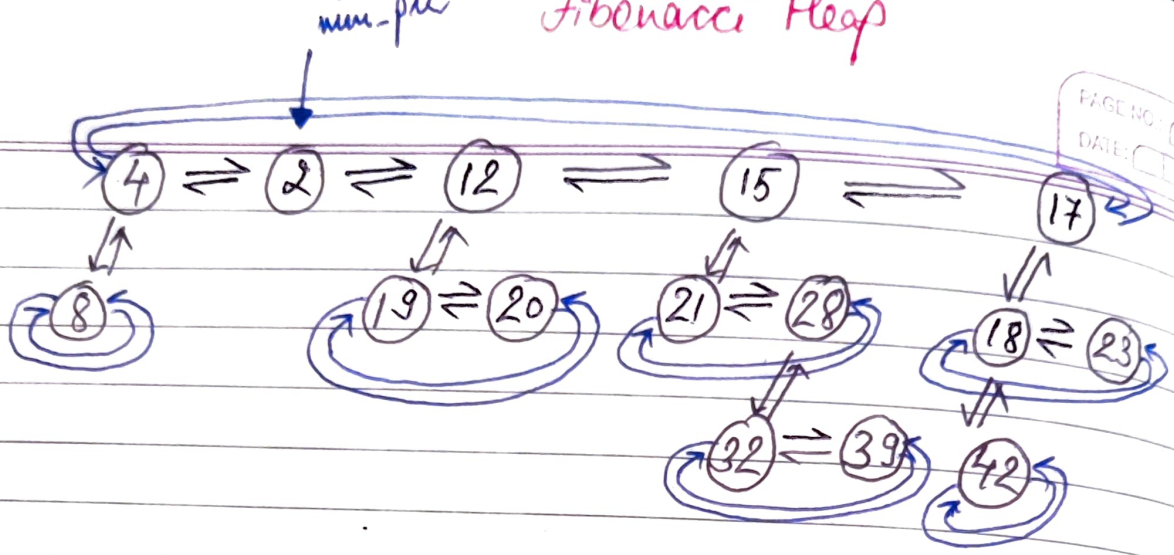
$\phi_{new} = t(H) + 2m(H)$
$= 5 + 2*3 = 11$
$\phi_{old} = 4 + 2*3 = 10$

$\Delta\phi = 1$

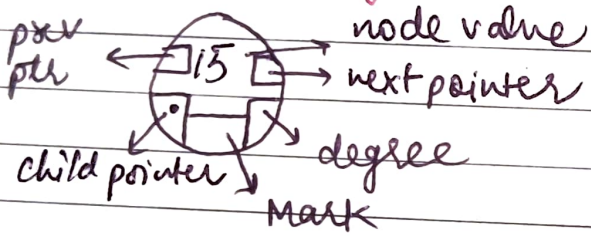
$C_{i_{insert}} = C_i + \Delta\phi = 1 + 1 = 2$

Amortised cost of insert = 2 = O(1)

Fibonacci Heap



Structure of node



Root list will always be unmarked

Union/Merge of two heaps. O(1)

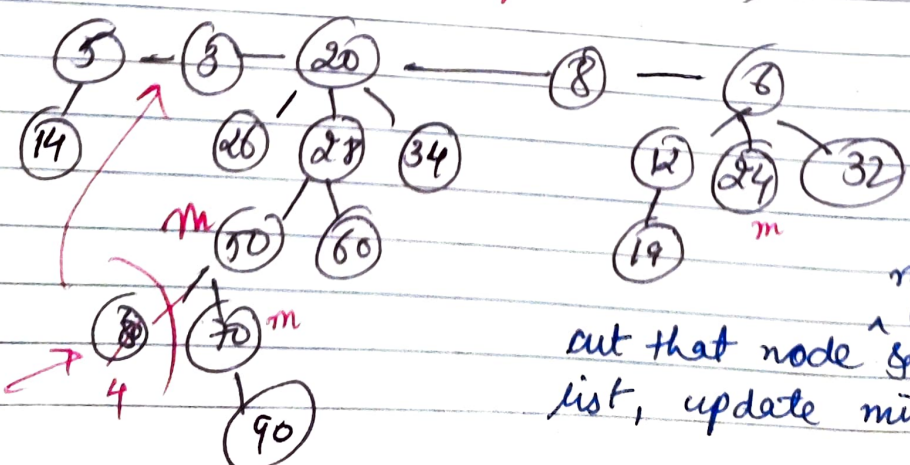
Let $\phi(H_1) = x_1$, $\phi(H_2) = x_2$

TC of Concatenation of 2 doubly linked list $\simeq O(1)$

just concatenate both heap and update min-ptr

Decrease the key (80, ~~80~~ 4) → Cascading cut

(old, new)

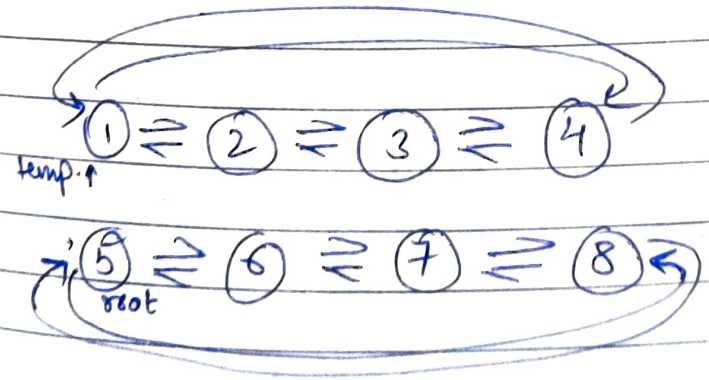


mark parent (if unmark)

cut that node & add to root list, update min-ptr

if parent already marked, cut the parent

Amortised cost for cascading



temp → prev → next = root

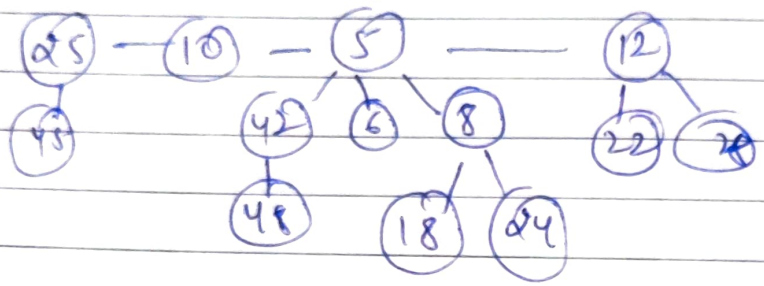
root → prev → next = temp

Date
22nd feb 2024
Thursday

PAGE NO.
DATE

Fibonacci heap

Extract-min ——— $O(\log n)$

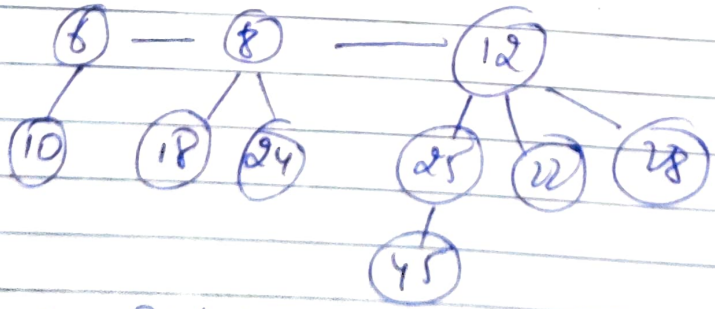


- Delete min-ptr & update min-ptr
- All child of deleted node will attach to root list
- Again, traverse root & update min-ptr

Array of pointers :-

store the pointer of root having i child on i^{th} position.

merge the trees with same degree using min-heap property



-: Consolidation Done :-

size of array = $(\log n) + 1$

1st step: Extract min – delete min node and add its children to root list

2nd step: $\hat{c}_i = c_i + \Delta\phi$ consolidation.

Actual cost = degree of deleted node. $= \log n$

$\phi = t(H) + 2 * m(H)$

$t(H)$: No. of trees; $m(H)$: marked nodes

Case I :- when all nodes are unmarked.

$\hat{c}_i = \log n + [\log(n) - 1]$

$= O(\log n)$

$6 + (2n-3) - (4 + 2n) \Rightarrow 6 + 2n - 3 - 4 - 2n$

$\Rightarrow -1$.

$\Delta\phi = -1$

$4 - (\log n)$

$\Delta\phi \rightarrow -2\log n$ if all nodes of del-node is marked

$\rightarrow 0$; Decrease.

Consolidation

$\hat{c}_i = t(H) + \log n - t(H)$

$\hat{c}_i = \log n$

Delete-key $(H, x) \longrightarrow O(1) + O(\log n)$

$\Rightarrow O(\log n)$

* All operation $\longrightarrow O(1)$

∴ except Delete-min & Delete-key $(\log n)$

Date

27th feb 2024
Tuesday

PAGE NO.:
DATE:

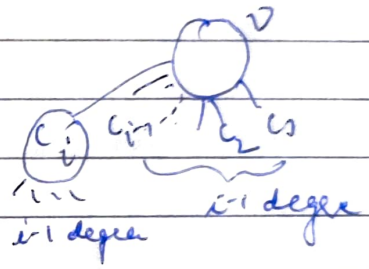
c_i is the i^{th} youngest child of a vertex v in a FH where $i > 1$

That means that v has

$c_1, c_2, c_3, \ldots c_{i-1}$

Show that rank of c_i must be $i-2$ at least

c_i has $i-1$ degree at the time it became at the i^{th} child of v



* Any tree of rank r in a Fibonacci heap has atleast $f(r+2)$ descendent (nodes)
  └ $(r+2)^{th}$ fibonacci no.

Let T_r be the smallest tree of rank r

Balanced BSTs

Date
29th Feb 2024
Thursday

Splay Trees

Perform splay operation on BST for restructuring the tree

① $O(n)$ – worst case for single operation
② $O(m \log n)$ → Amortised complexity for series of m operations

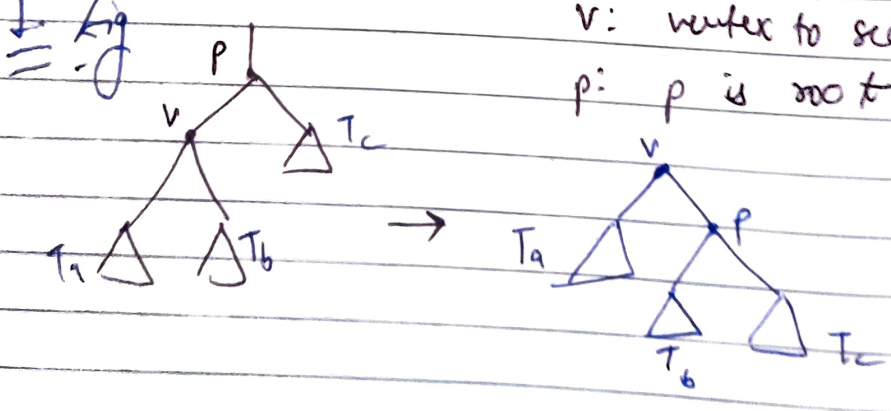
Principle of locality

Special locality
Temporal locality

Splay Trees

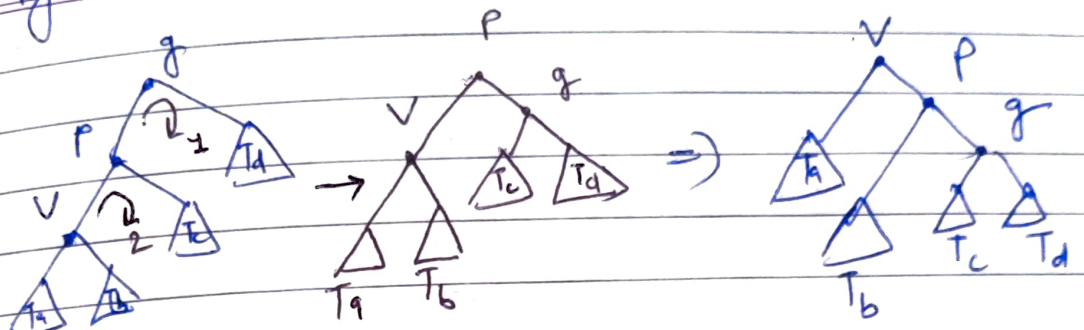
1. Zig – Single Rotation
2. Zig-Zig – Double rotation (same direction)
3. Zig-Zag – Double rotation (Different Direction)

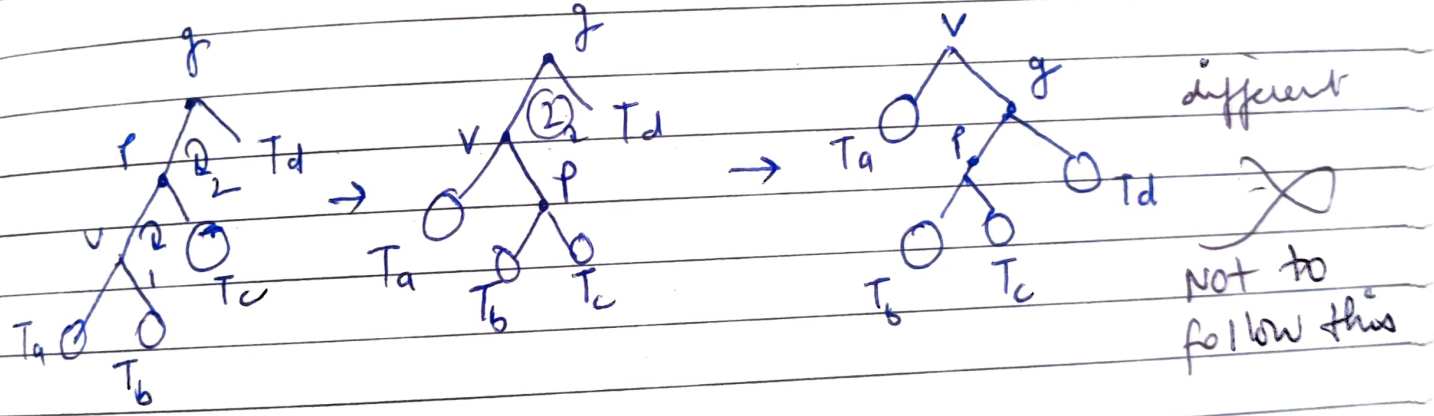
Zig



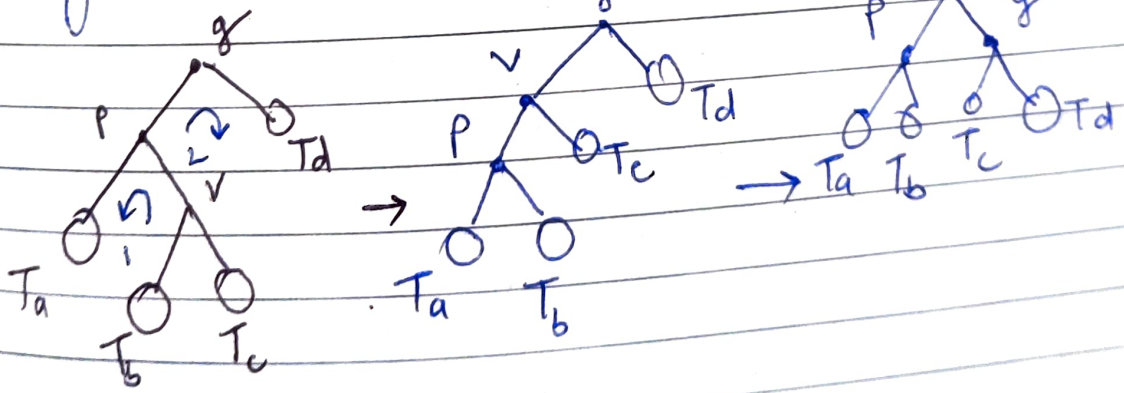
v: vertex to search
p: p is root

2. Zig-Zig - Double rotation



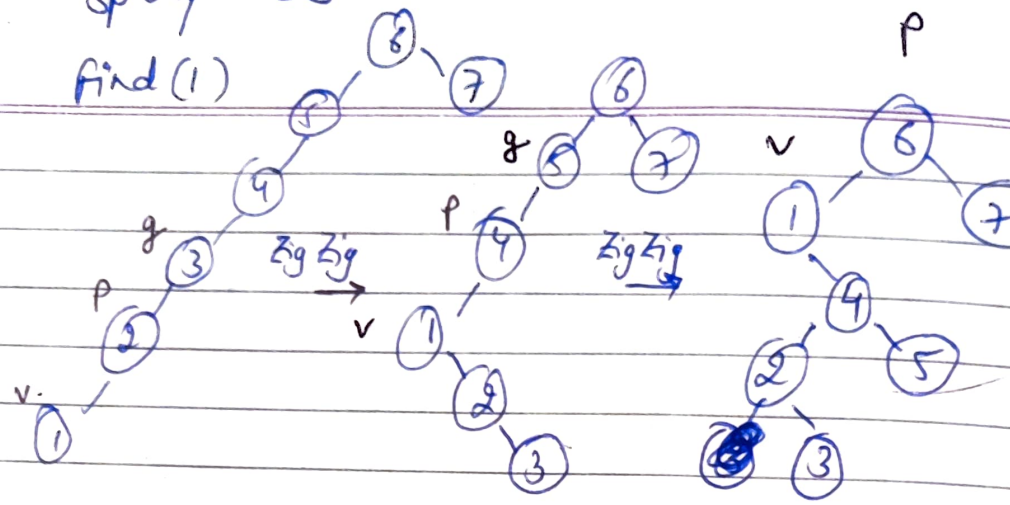


3. Zig-Zag Double rotation

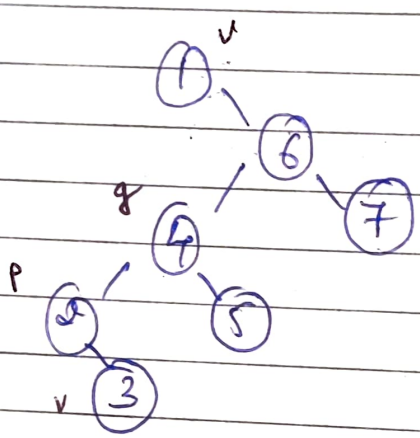


splay trees

Find (1)



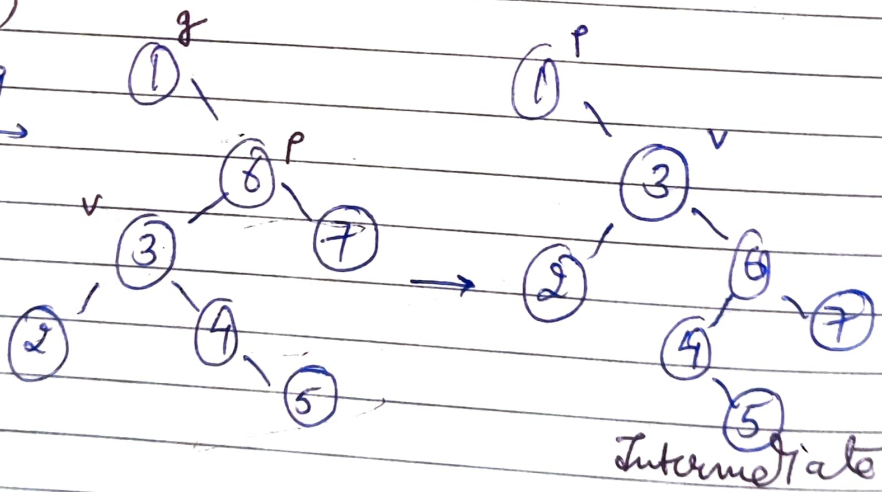
Zig

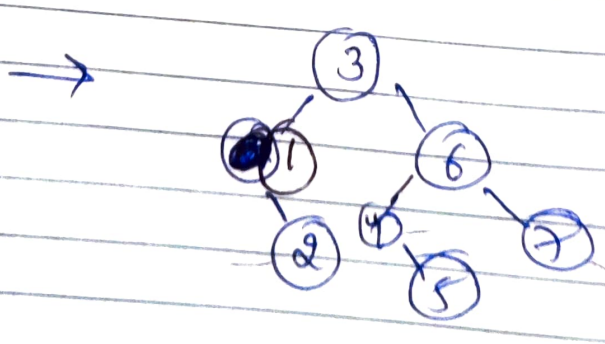


dup

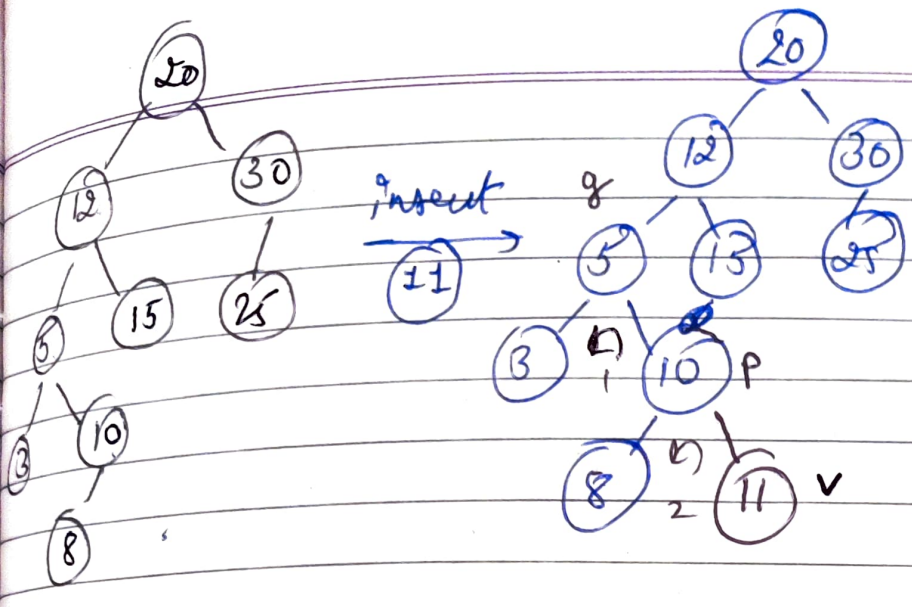
Find (3)

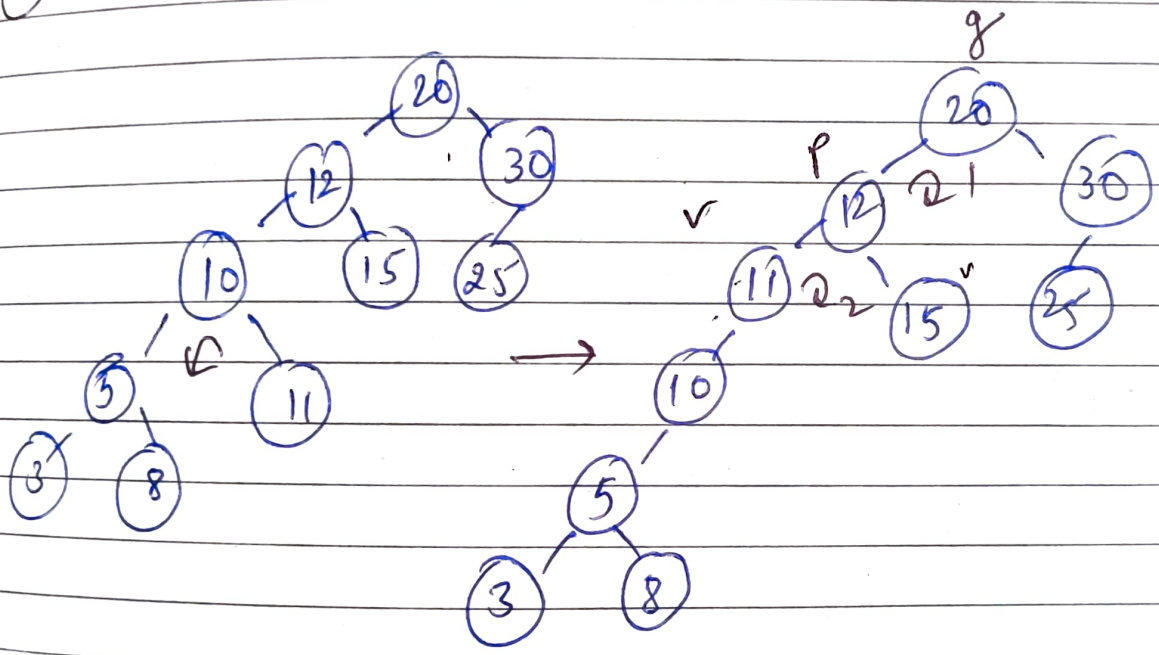
Zig Zag

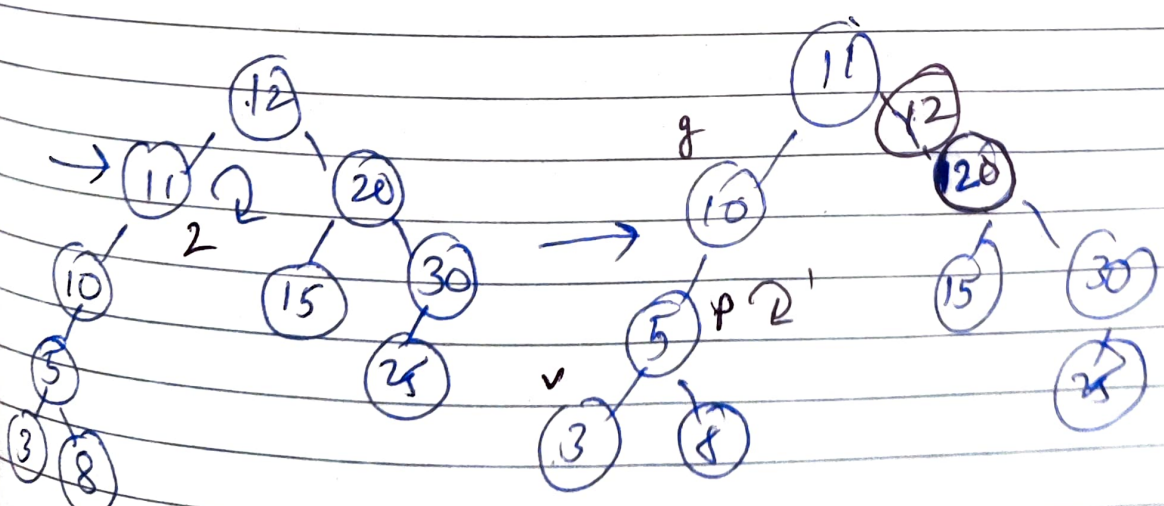




Insert (11) & Splay

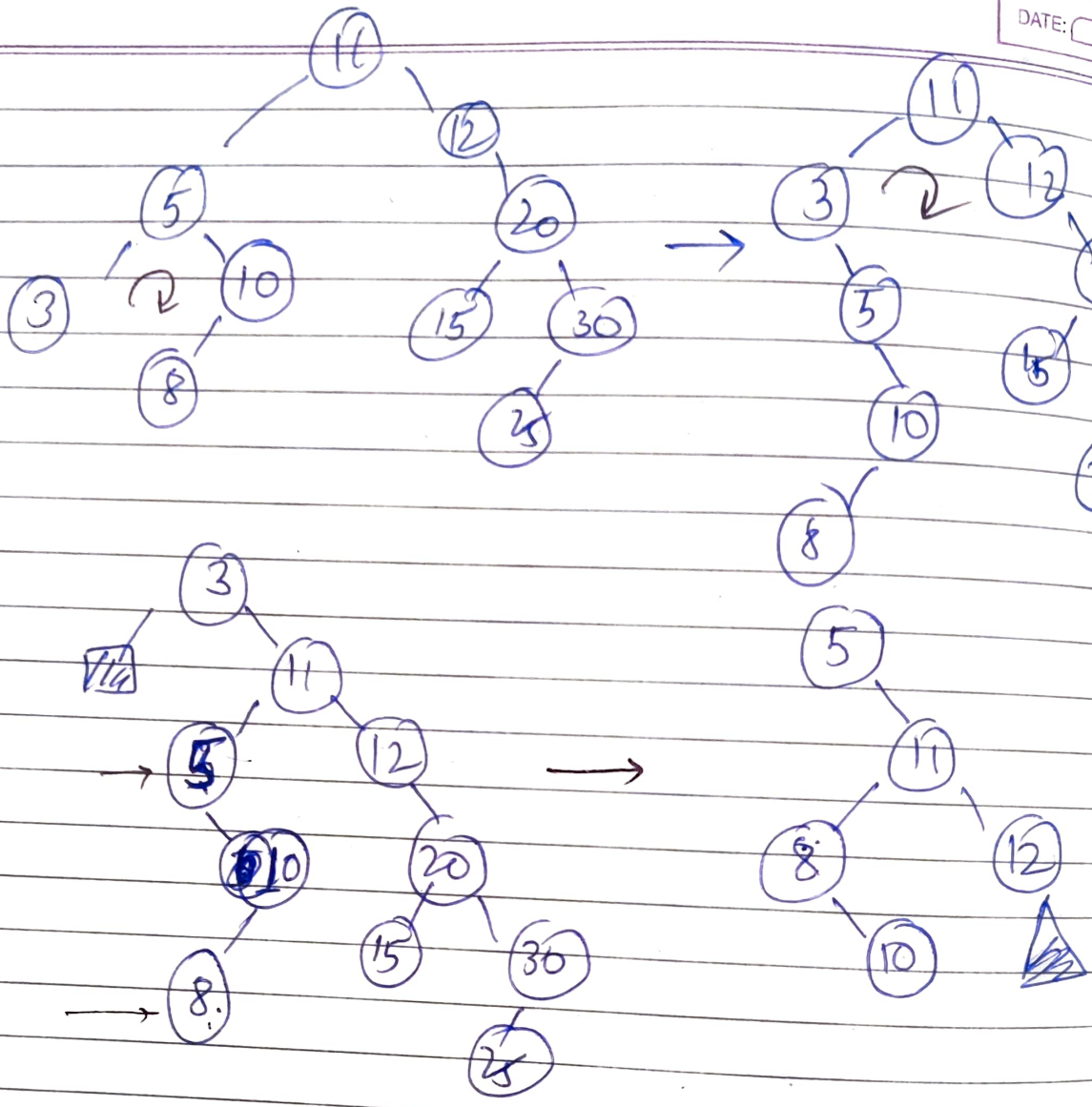






Deleting (3)

→ Splay

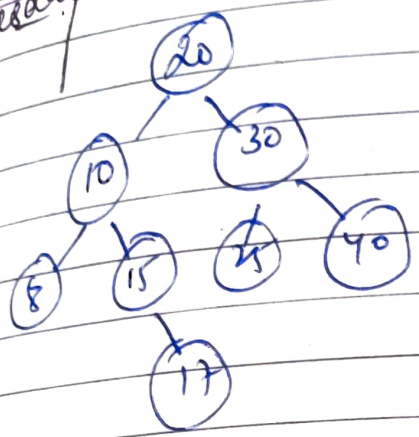


* for unsuccessful search, last visited node is splayed.

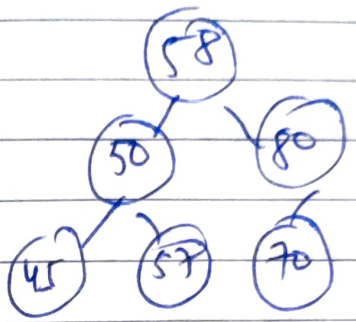
Date
5th Mar 2024
Tuesday

Join

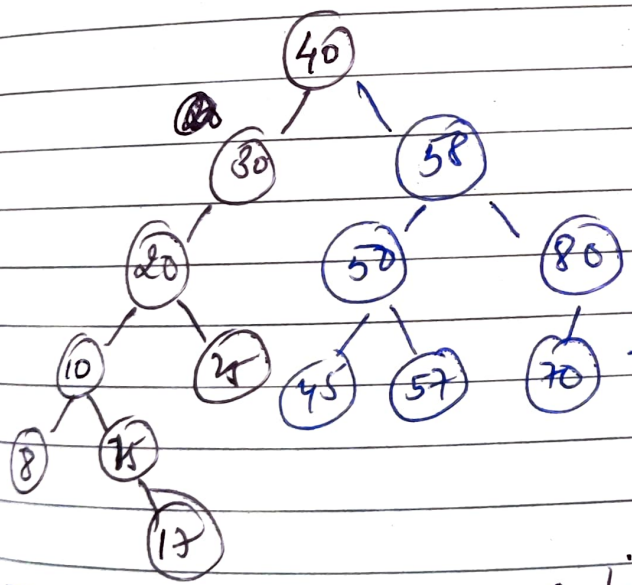
T_1



T_2



splay the max node of T1 -then,



attach T2 on right subtree of T1.

✓ DONE

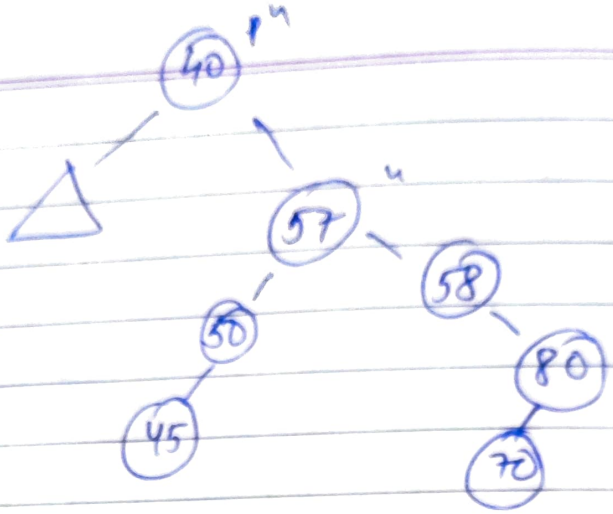
Split

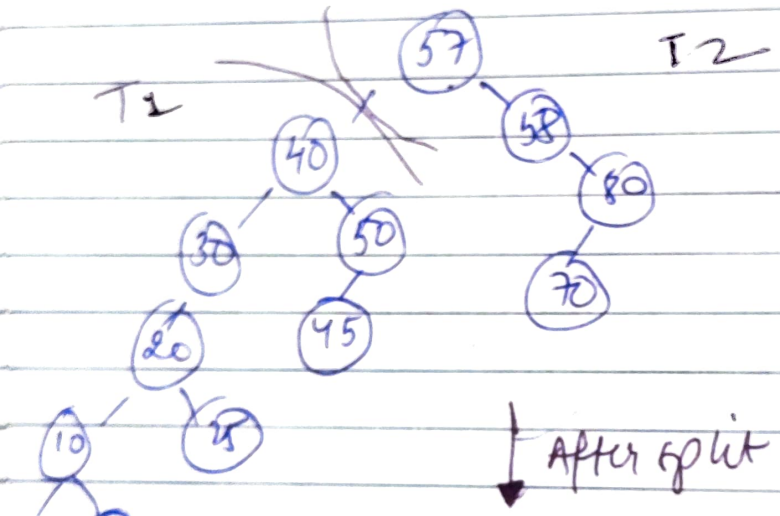
Split the tree at key 'k' = 50
(node ≤ 50 → single tree) & rest

Splay inorder successor of k = 50, & cut left subtree

OR

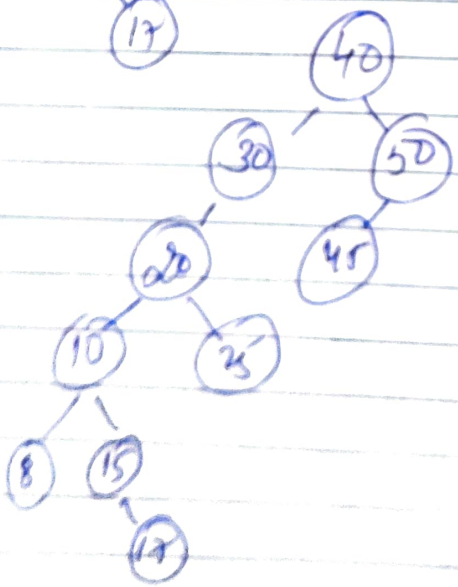
Splay k = 50, & cut rightsubtree



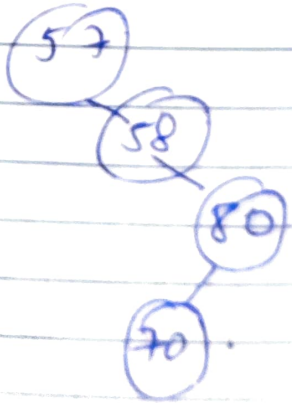


After split

T1



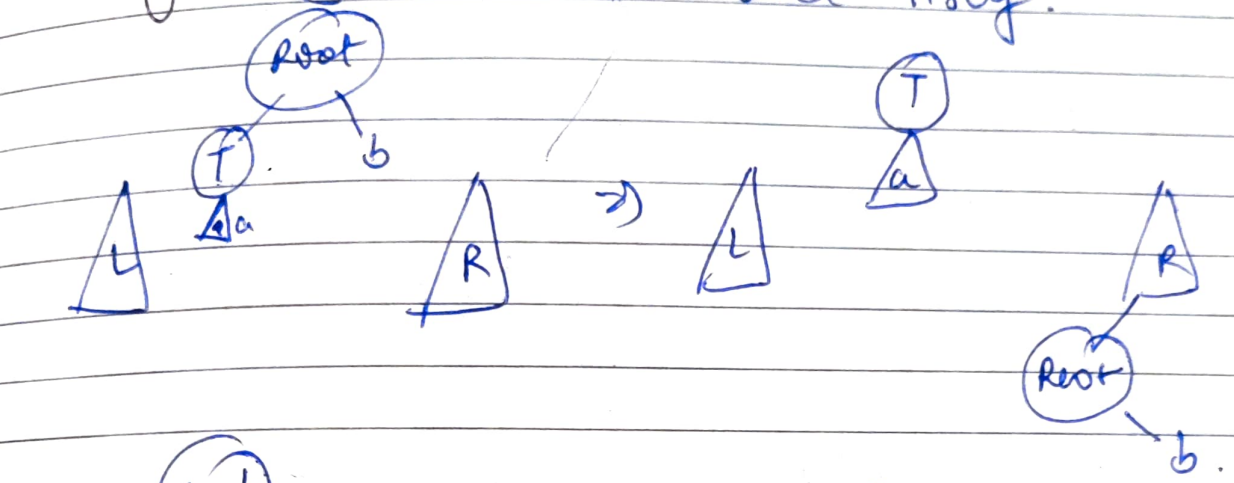
T2

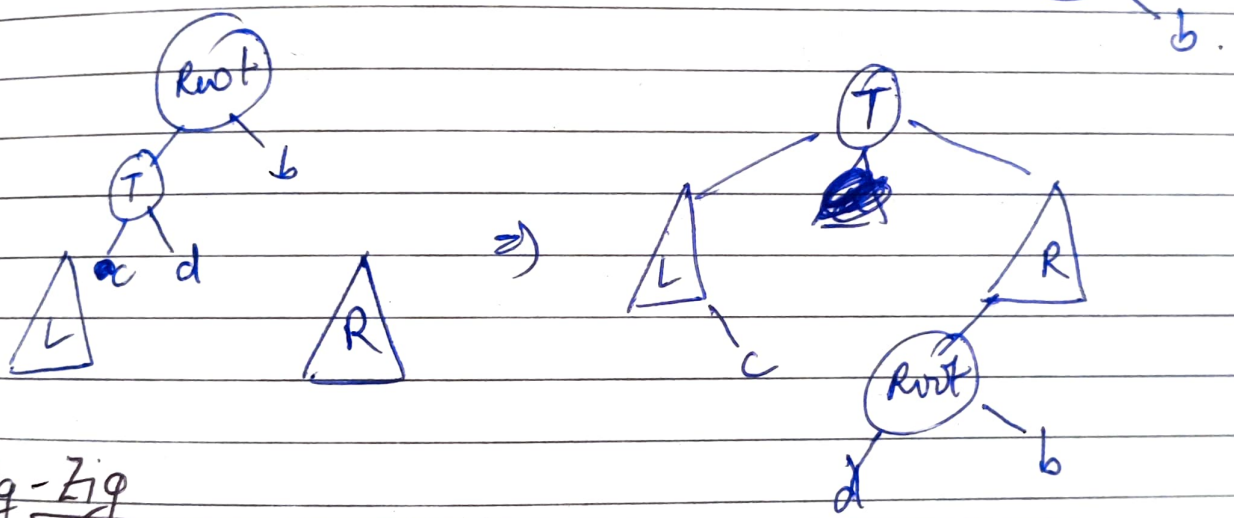


Bottom-up splaying occurred

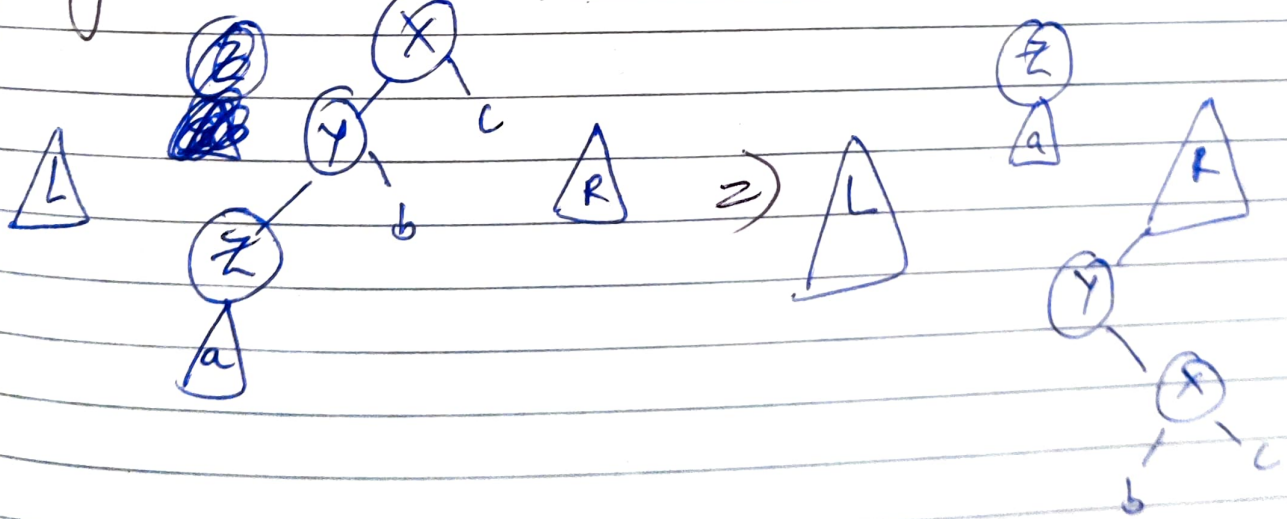
Top-down splaying

* Zig operation: When node to be search is on next level itself.

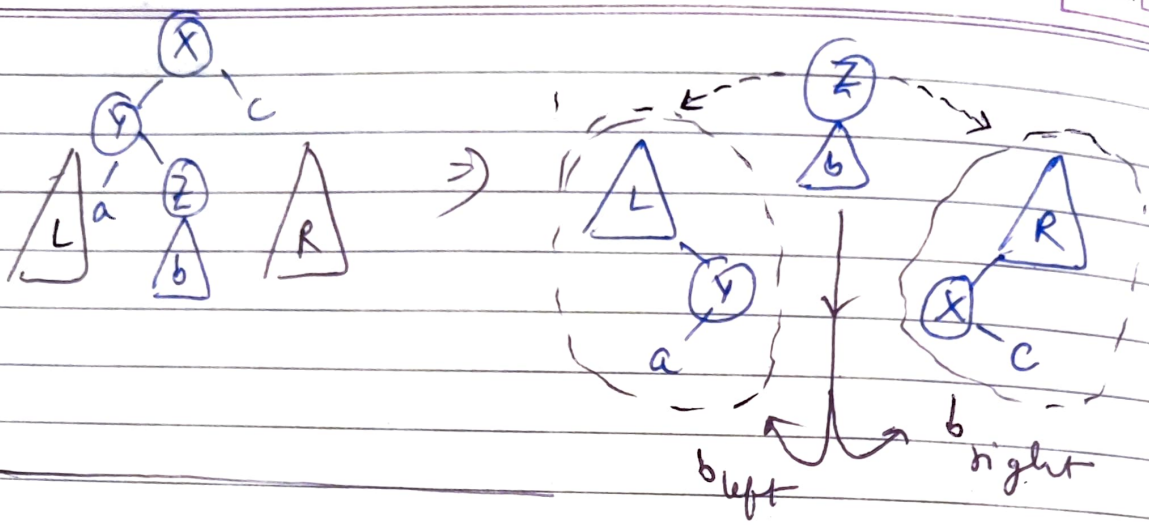


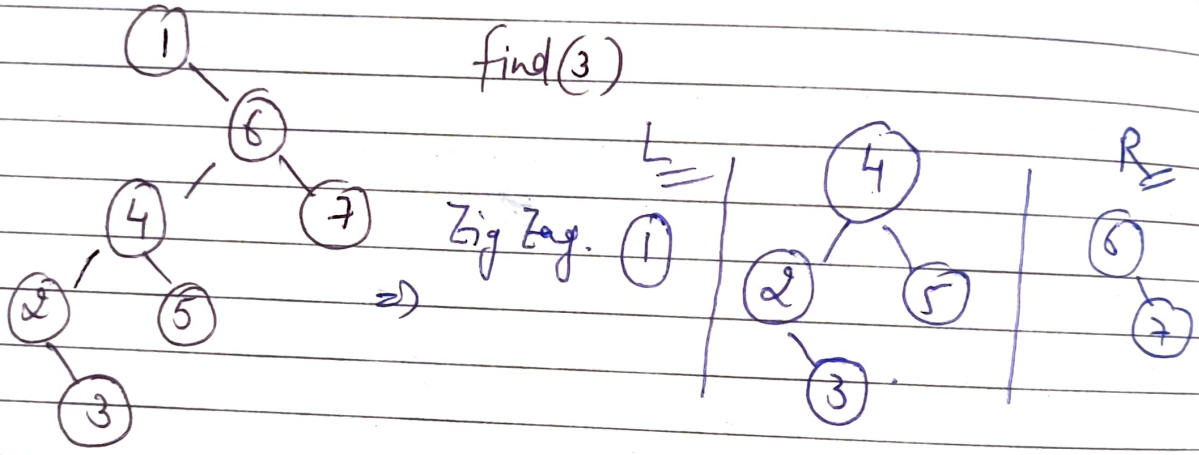


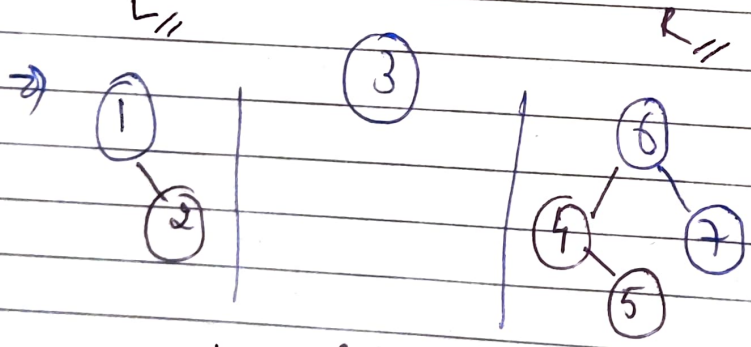
Zig-Zig



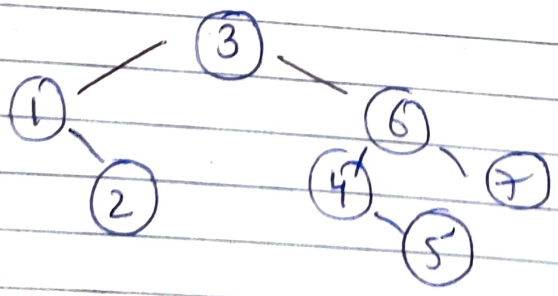
Zig-Zag :-

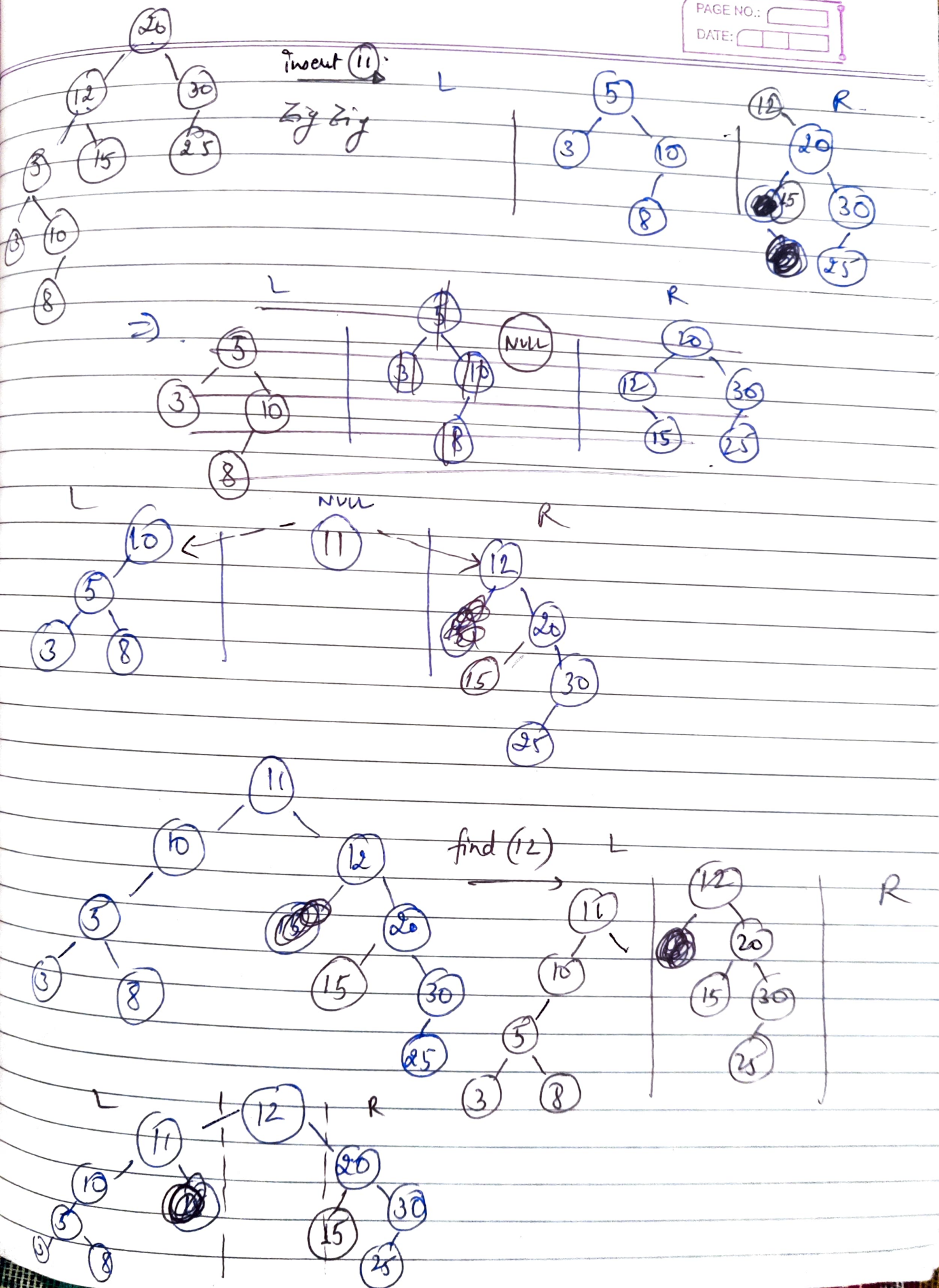






→ combine :-




insert (11)
zig zig
L
R
NULL
find (12)