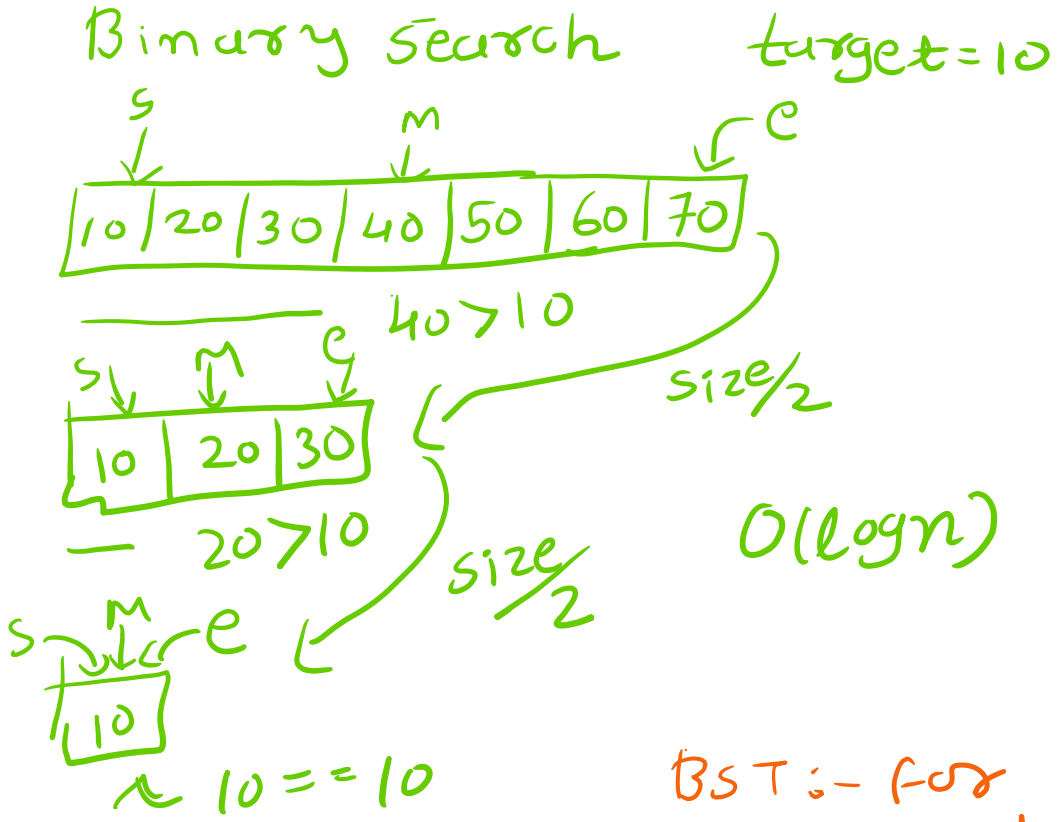
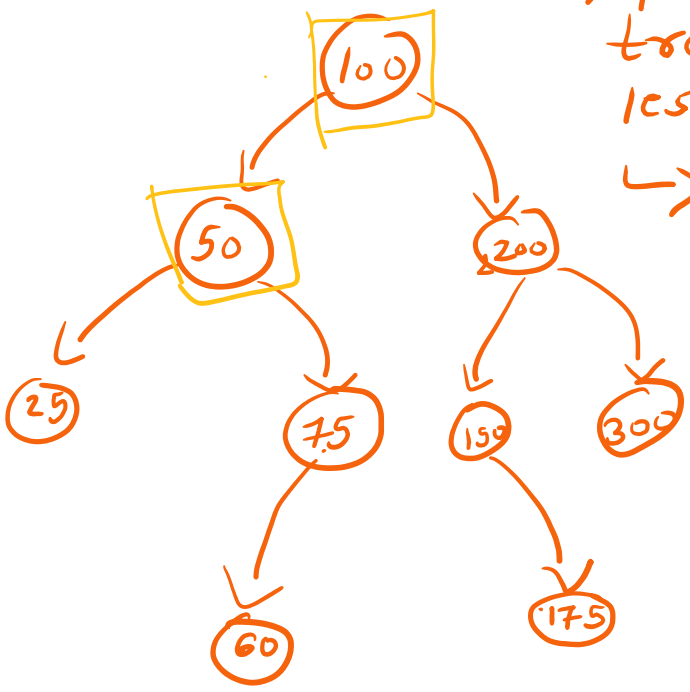
Binary Search Tree (BST)



BST :- for every node

↳ left sub tree data less

↳ right sub tree data are more.



root > data

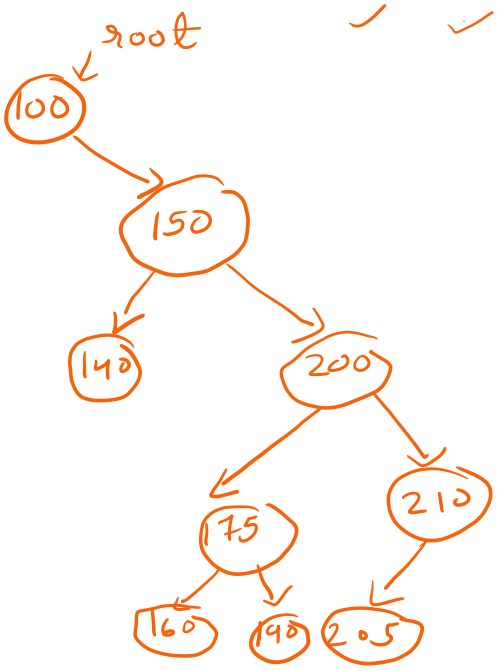
> (left -> sub tree data)

(left -> subtree data)

root -> data < (right -> subtree data)

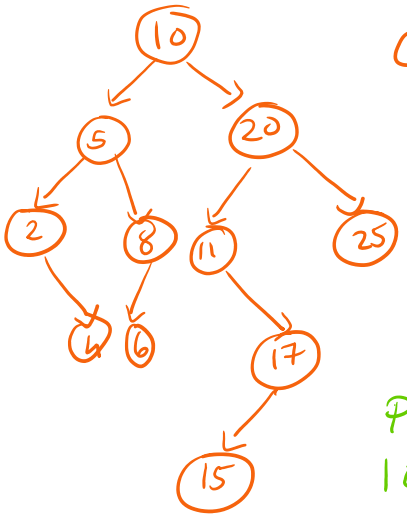
i/p :- 100 150 200 175 160 140 210
205 190

root = NULL



Inorder
↳ sorted number
always

i/p 10 20 5 11 17 2 4 8 6 25 15



Creation of BST

Postorder
LRN

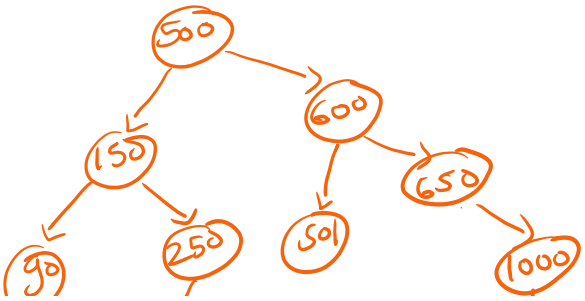
Inorder LNR

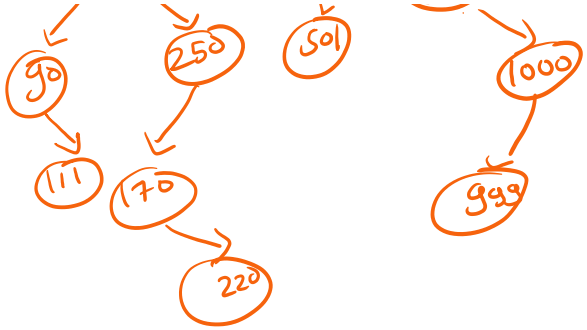
2 4 5 6 8 10
11 15 17 20 25

POC NLR

10 5 2 4 8 6 20 11
17 15 25

i/p 500 150 250 600 650 170 90 220
501 1000 111 999

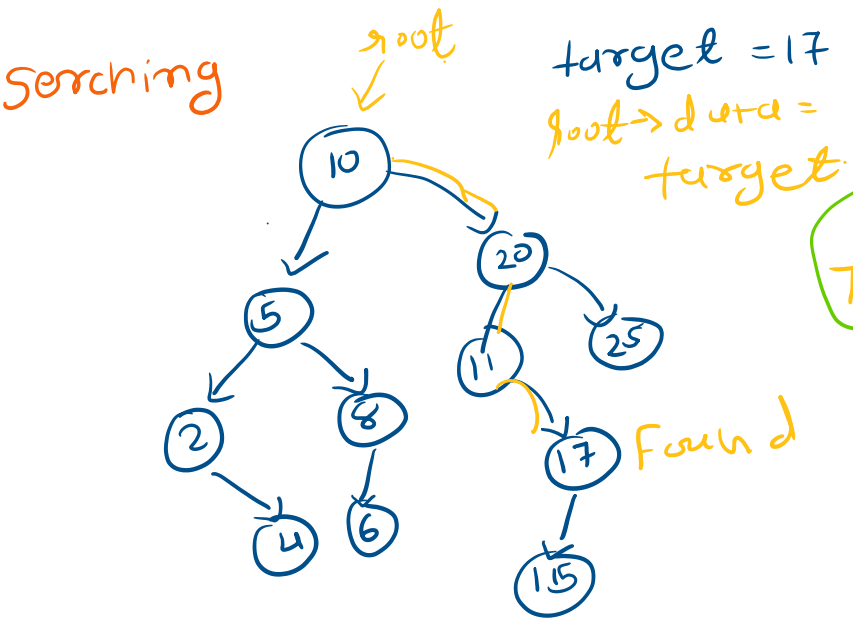



90
250
501
1000
111
170
999
220

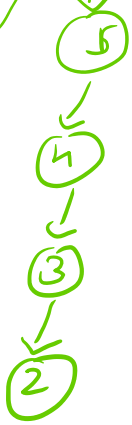
Rukna kab hai

(-1) i/p

Searching



worst case
O(n)



average case
Tc = O(H)
↳ O(log n)

Minimum value

#1 inorder

extra store

#2

કોઈ પણ Tree માં left-left પર બસ left કરતા જાવ તો

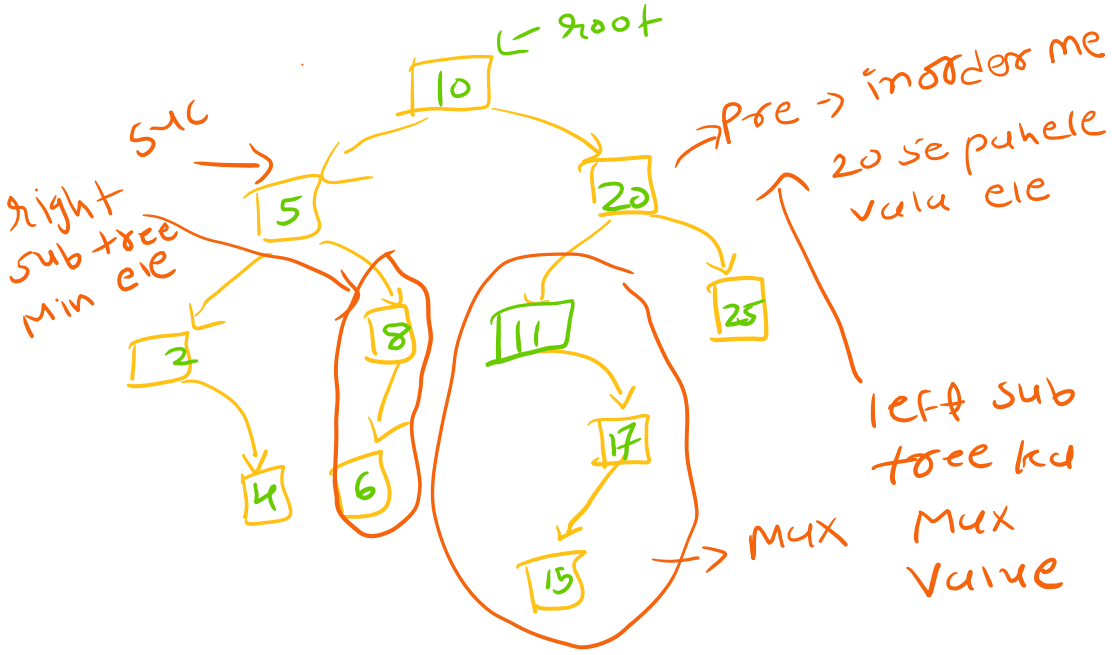
જાવ અંત left જ્યાં ગયા થાય તે
min.
Max માં Right - Right

29 March 2024 13:09

left ST + Max

** Inorder predecessor / successor

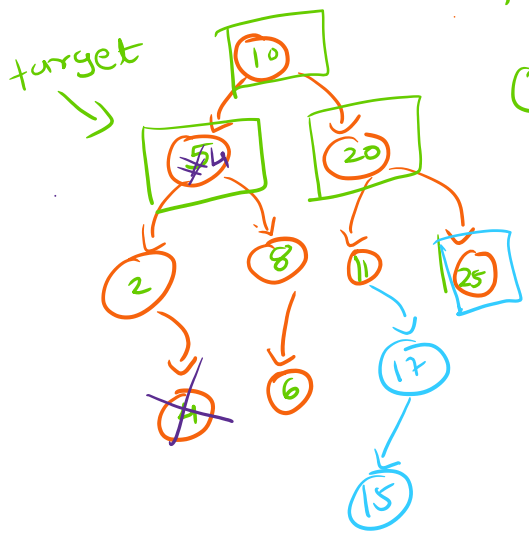
right ST + Min



2 4 5 6 8 10 11 15 17 20 25

Suc Pre

450 Deletion in BST

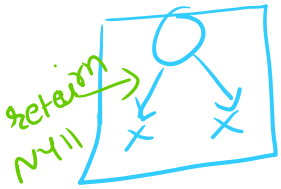


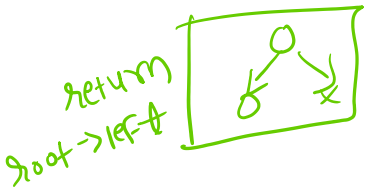
target = 25

① Find 25

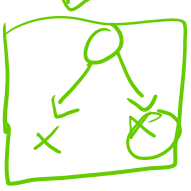
② delete 25

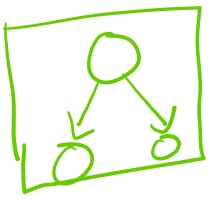
4 case





return root->right



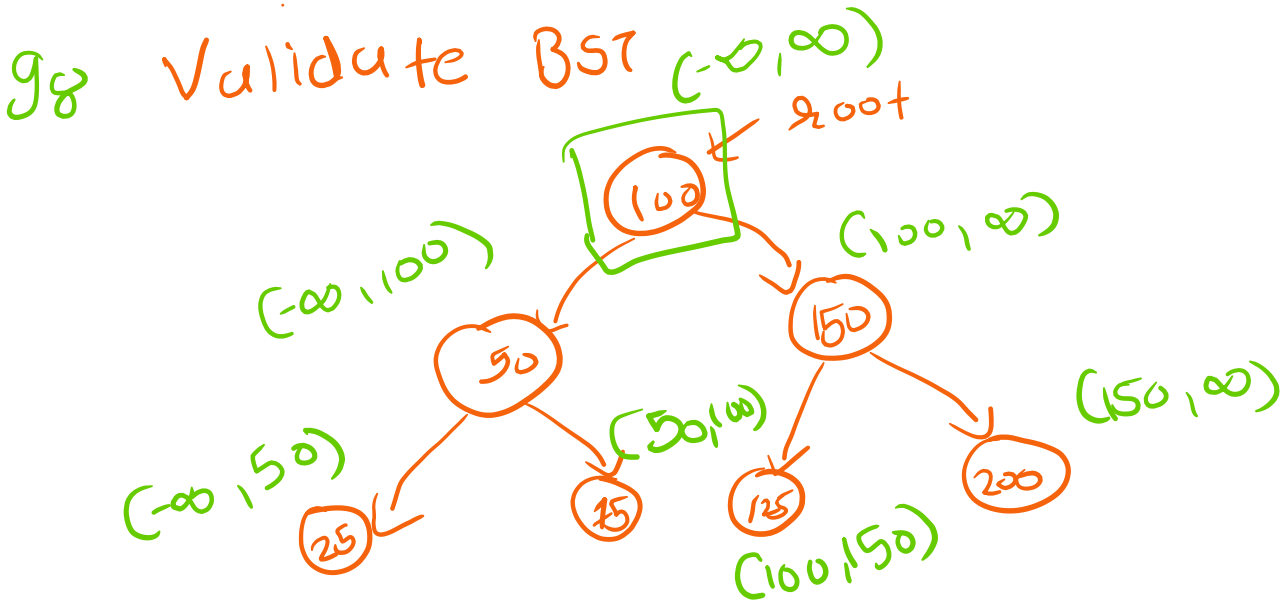


root->data

I.P I.S

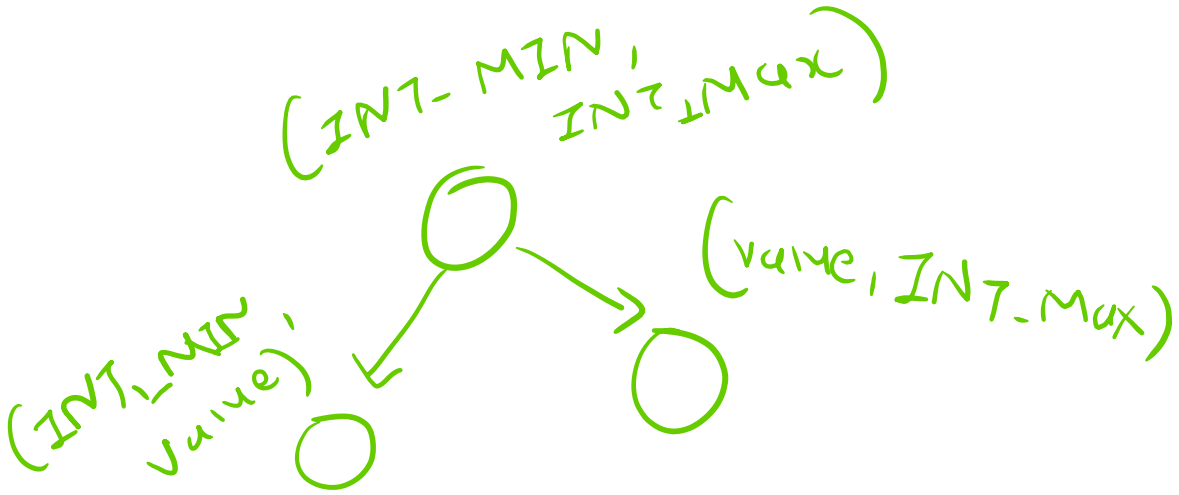
Height / Diameter of BST

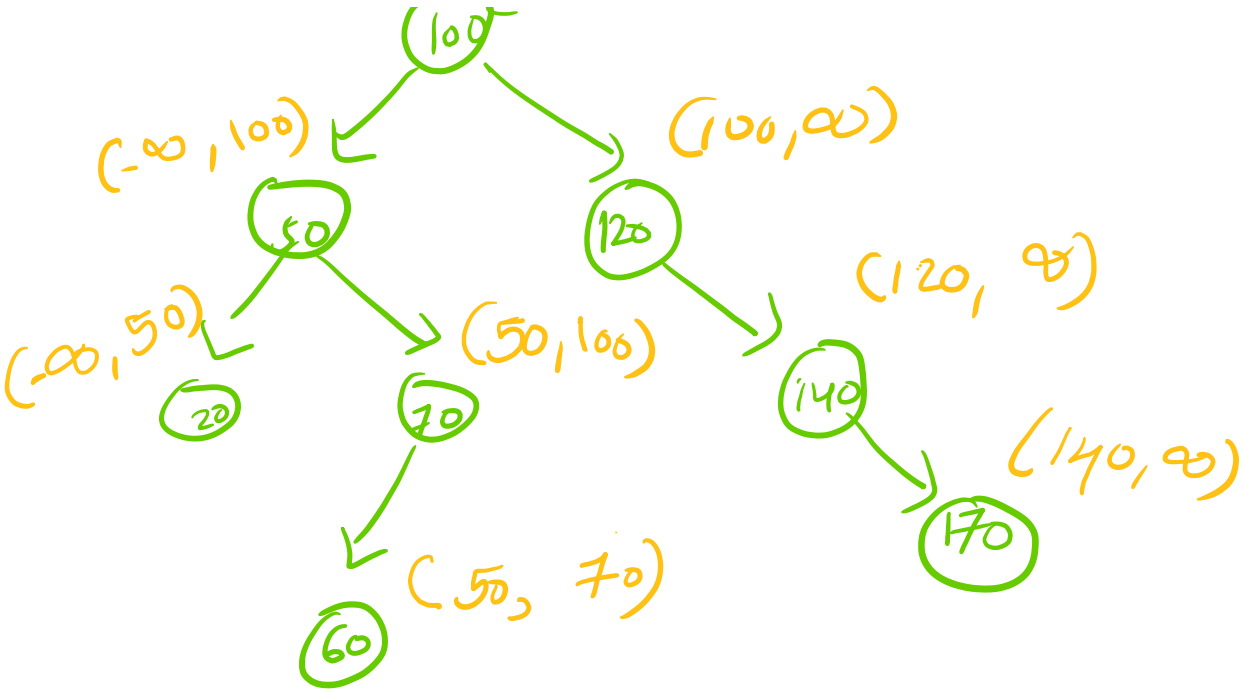
98 Validate BST



1. inorder
   - sorted ⇒ ? O(n)

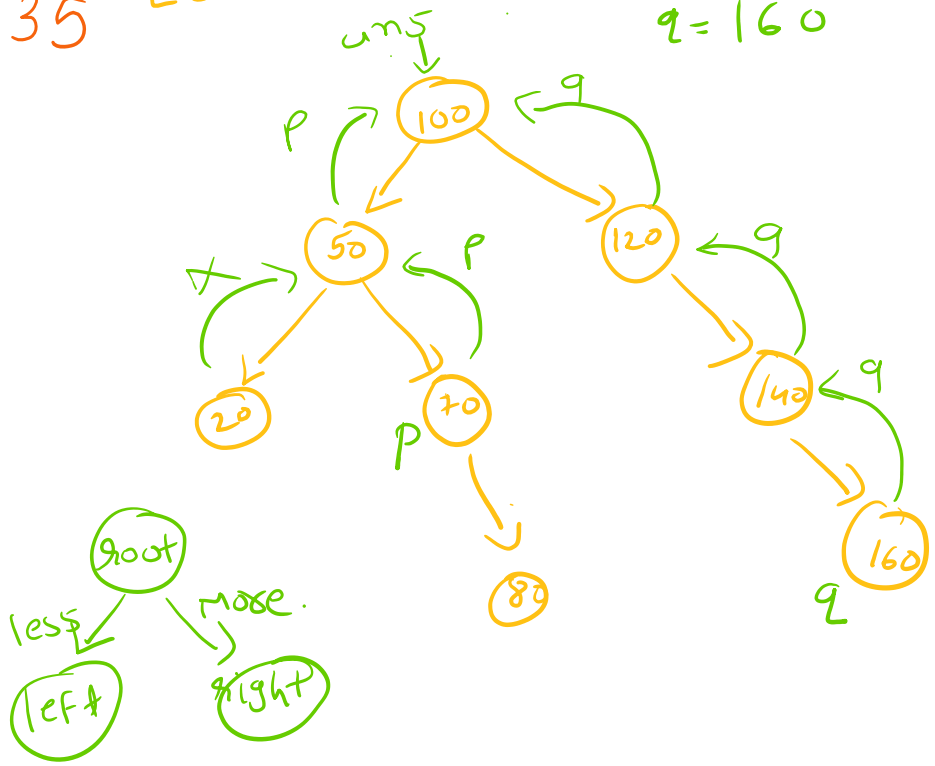
2. Recursion




100
(-∞, 100)
(100, ∞)
50
120
(-∞, 50)
(50, 100)
(120, ∞)
20
70
140
(140, ∞)
170
(50, 70)
60

235 LCA OF BST

p = 70
q = 160

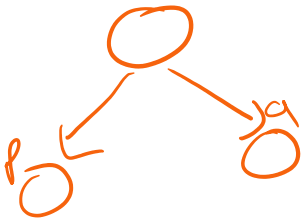


p < root->data
q < root->data

↓
left call

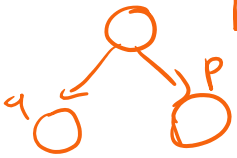
p > root->data
q > root->data

↑
right call



p < root->data
q > root->data

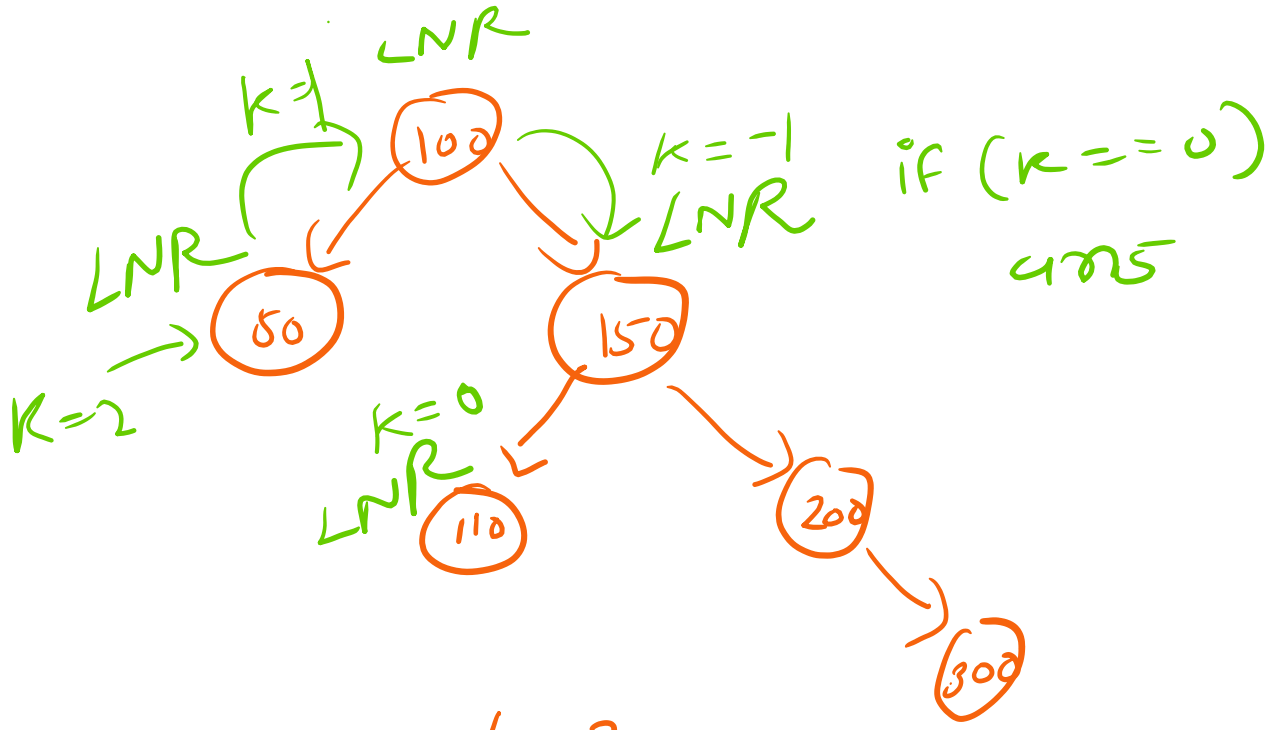
↑
ans = root



p > root->data
q < root->data

↑
ans = root

230 k^{th} smallest value on BST



if (k==0) ans

k=3

inorder TC = O(n) SC = O(n)

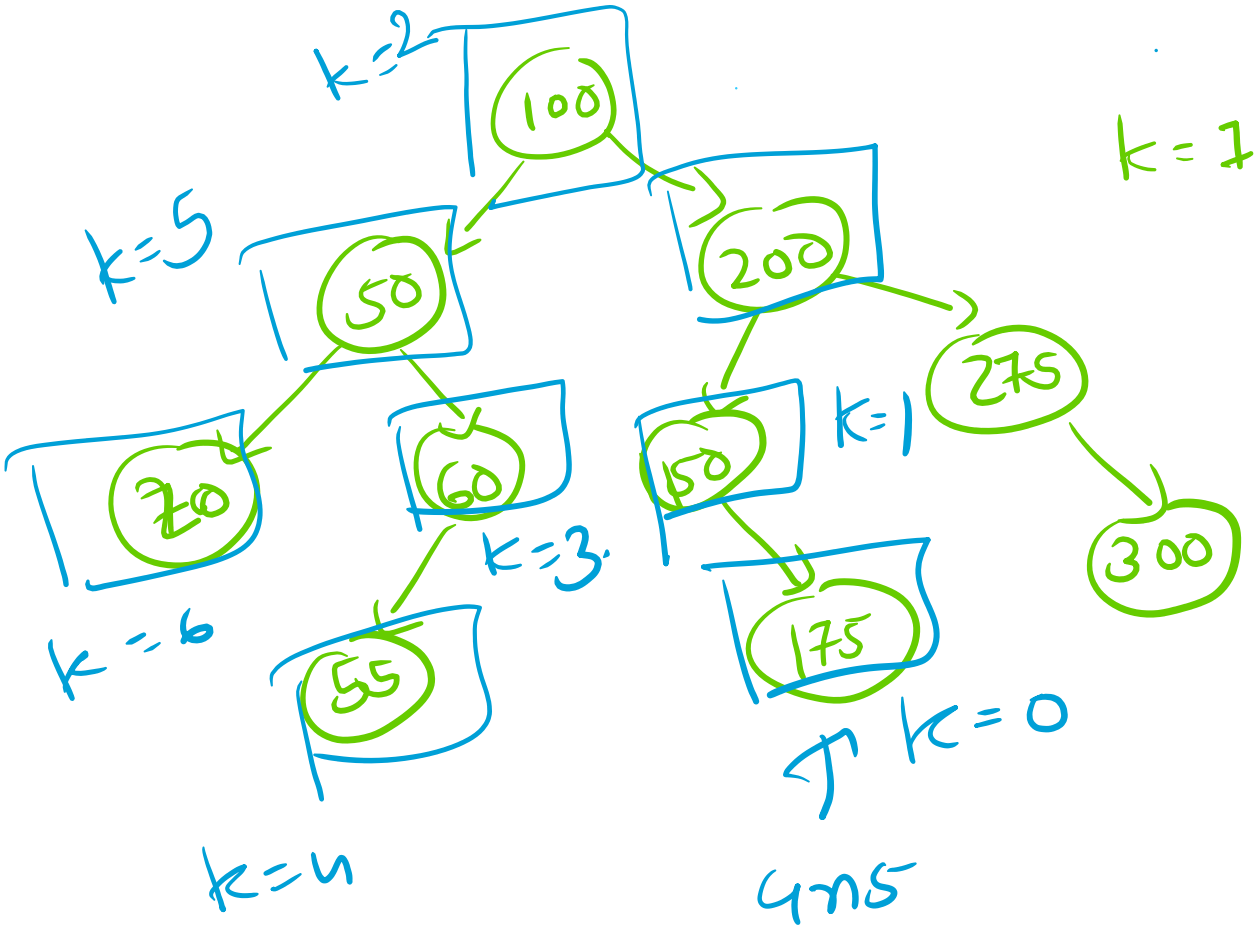
50 100 110 150 200 300

↑ ans

inorder traversal

LNR → right

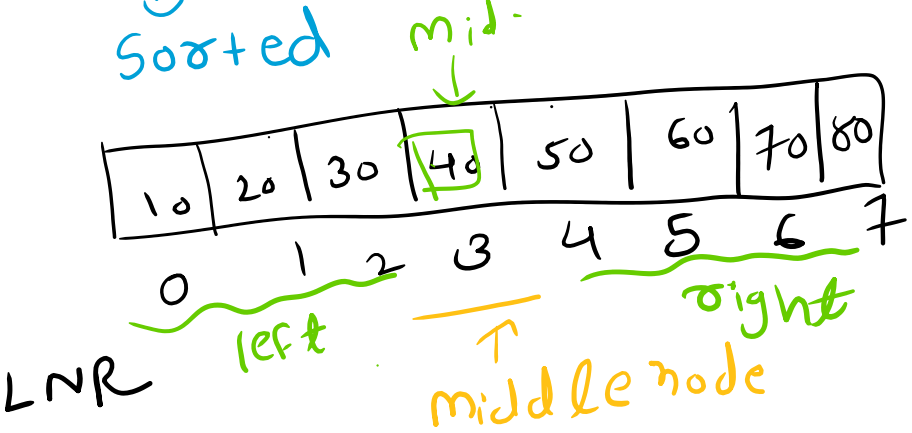
↳ left root


k=2
100
k=1
k=5
50
200
275
20
60
150
k=1
300
k=6
k=3
55
175
k=0
k=4
yes

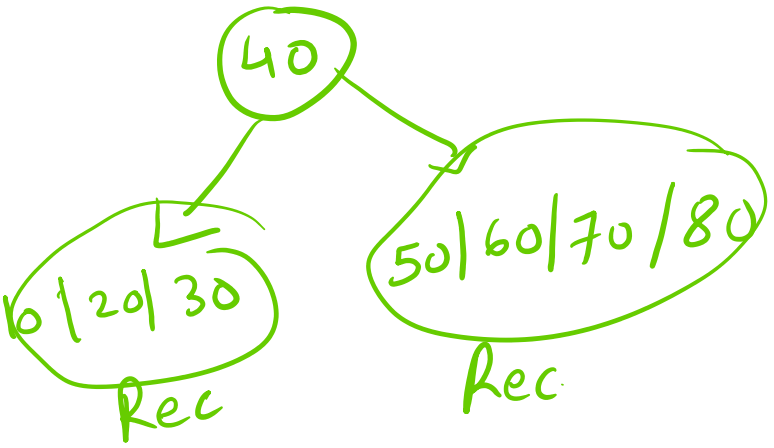
Inorder trave → BST create

↓

Sorted



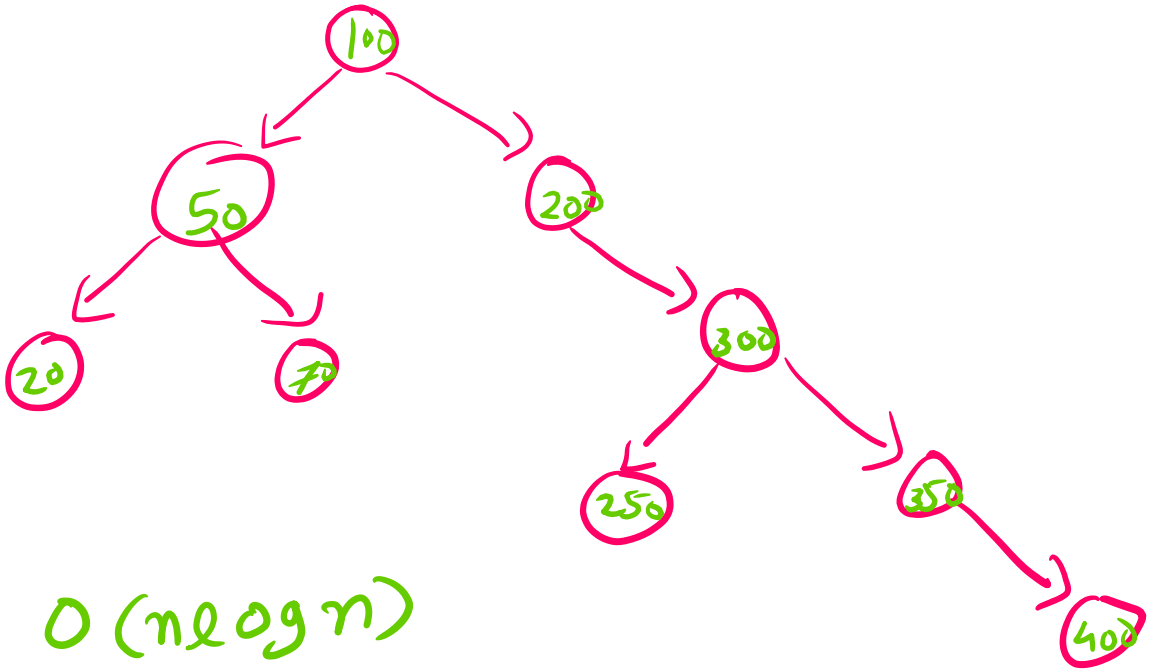
S=0, e=7 → mid = (0+7)/2 = 3



Convert a BST into a balance bst

653 2-sum

target = 320

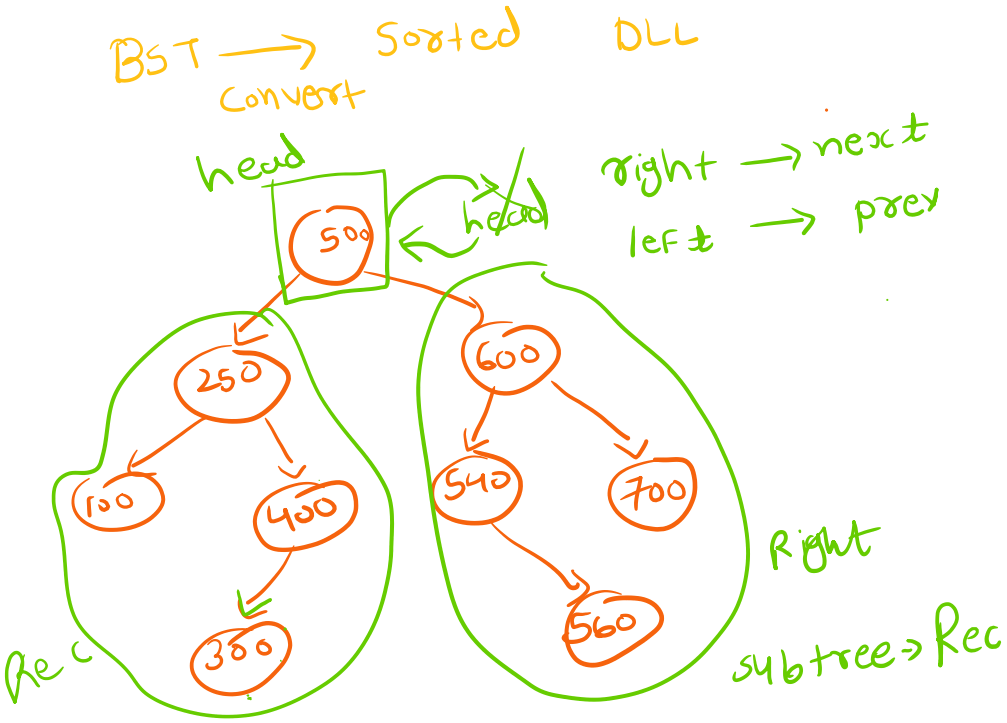


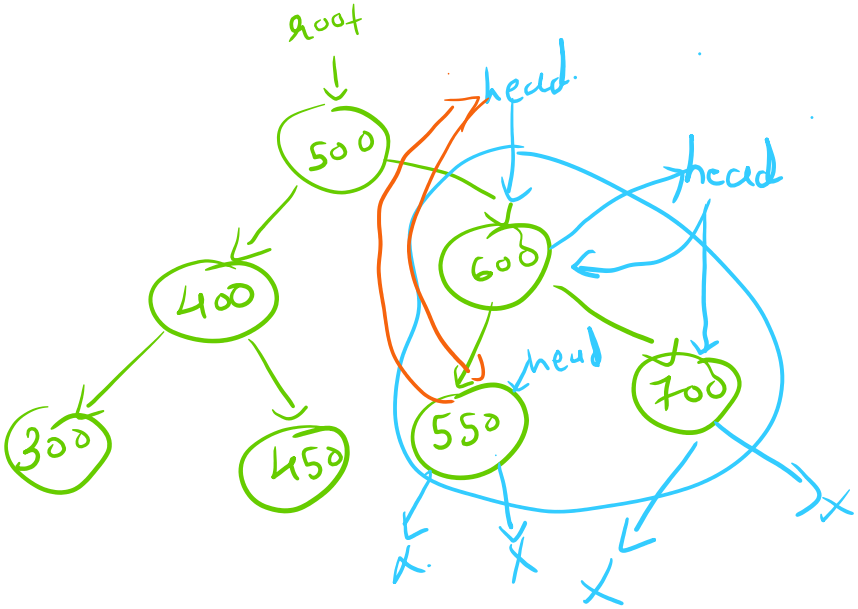
O(n log n)

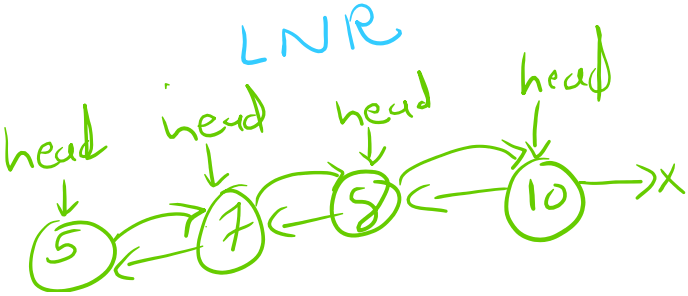
1 node → only left node check

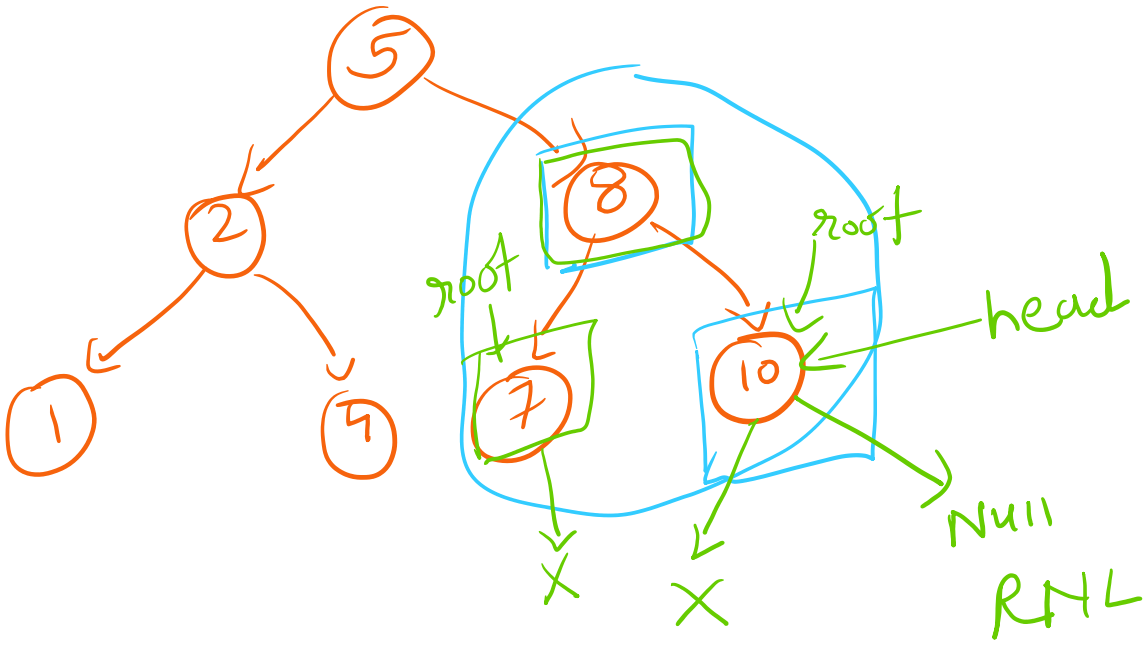
20, 50, 70, 100, 200, 250, 300, 350, 400
0 1 2 3 4 5 6 7 8

5 e e e


BST → Sorted DLL
convert
head
500
head
right → next
left → prev
250
600
100
400
540
700
300
560
Right
Rec
subtree → Rec


root
500
head
head
600
400
head
700
300
550
450


LNR
head
head
head
head
5
7
8
10



head == NULL

head == root

head != NULL

root → right = head

head → left = root

algorithm

① Right subtree

② root attach

root → right = head

head → left = root.

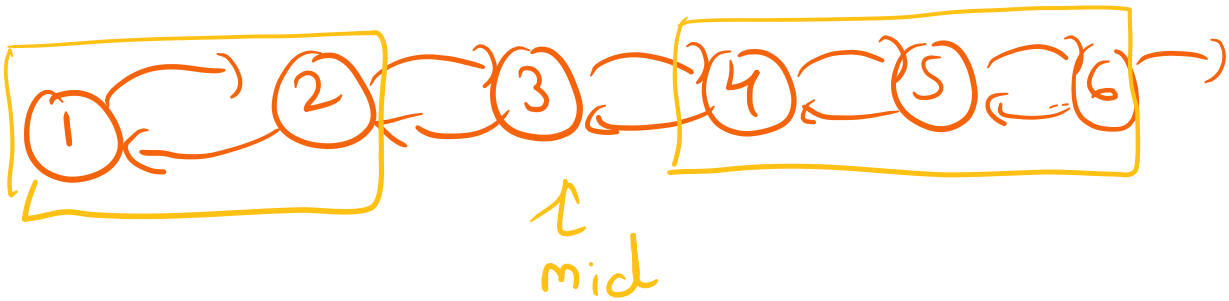
③ head attach

head = root

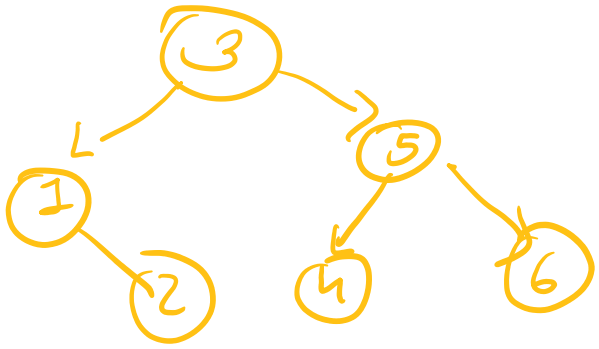
④ left subtree

Sorted LL ⟶ BST

i/p



o/p :



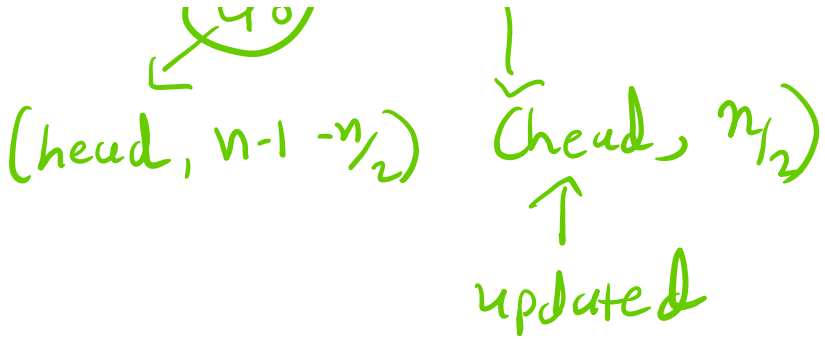
n-1-n/2 | mid | n/2

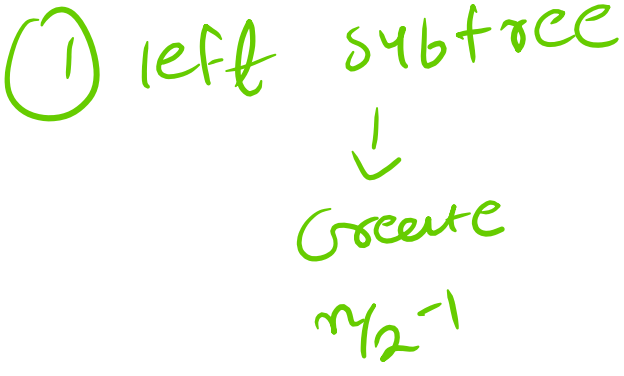
1

n nodes

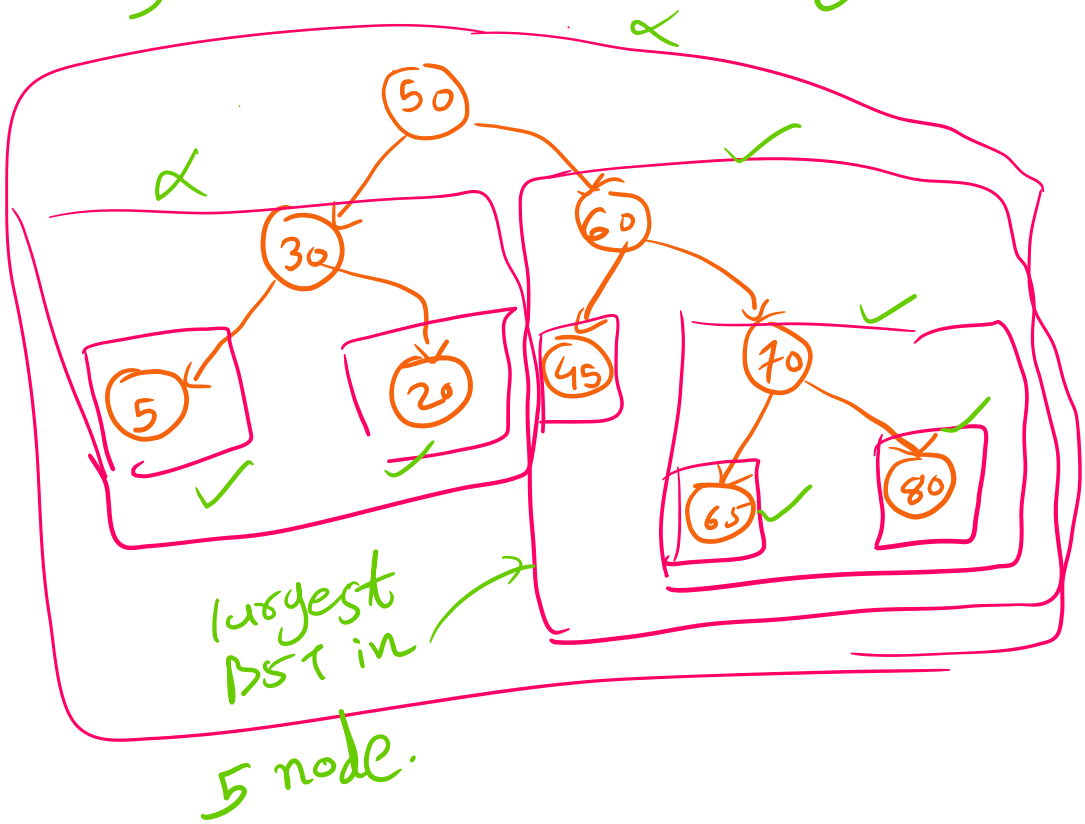
n=8

Big problem (head, n)

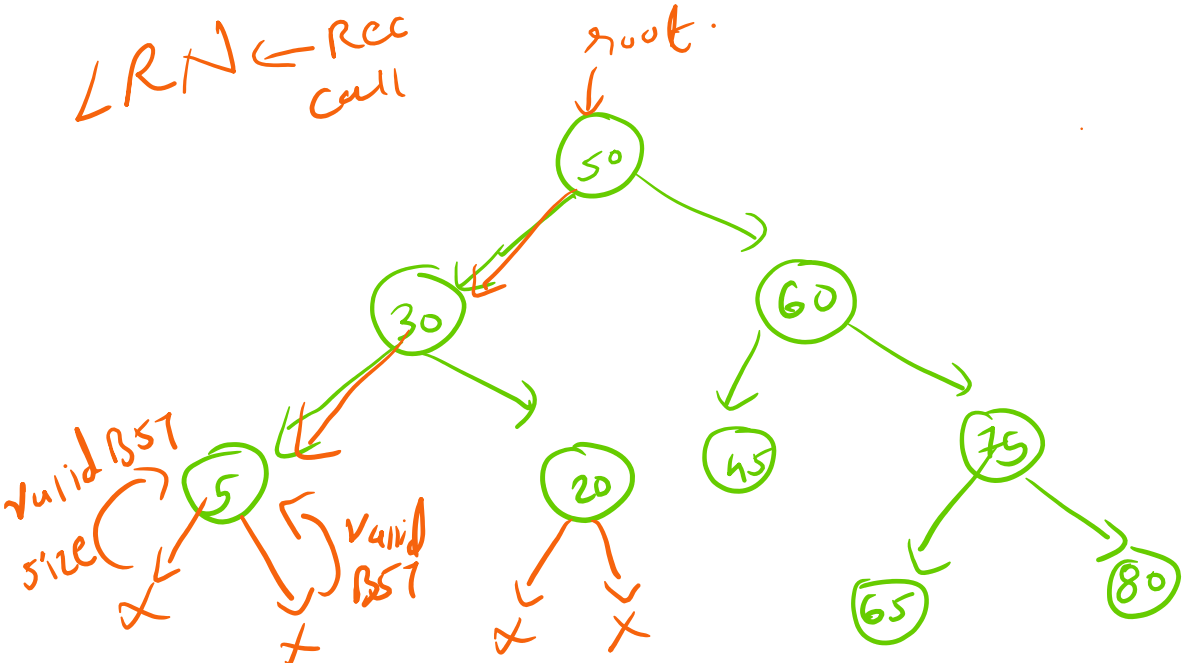

(head, n-1 -n/2)
(head, n/2)
updated

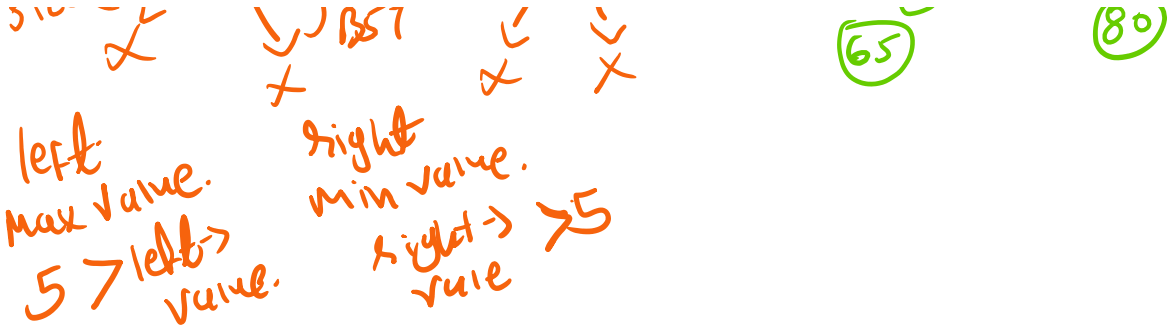

1 left subtree
create
n/2 -1

largest BST in a Binary tree

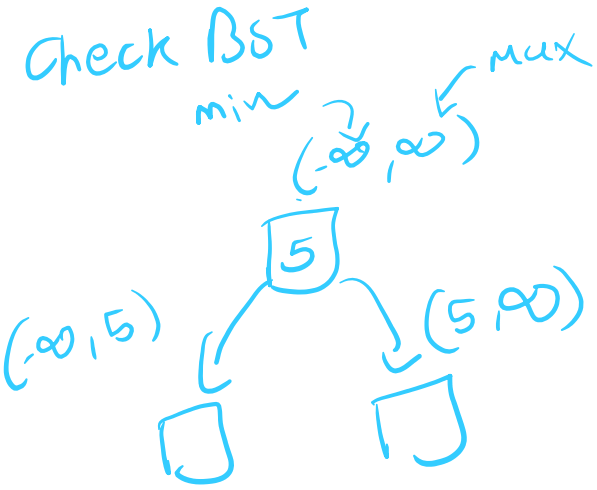


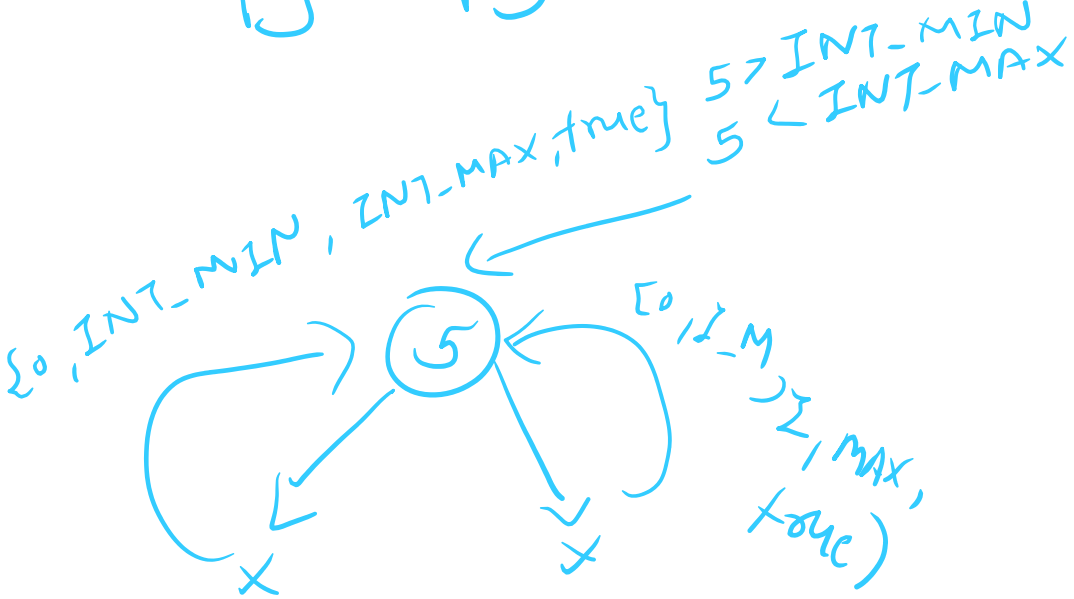
#1 inorder traversal sorted subset find max find.

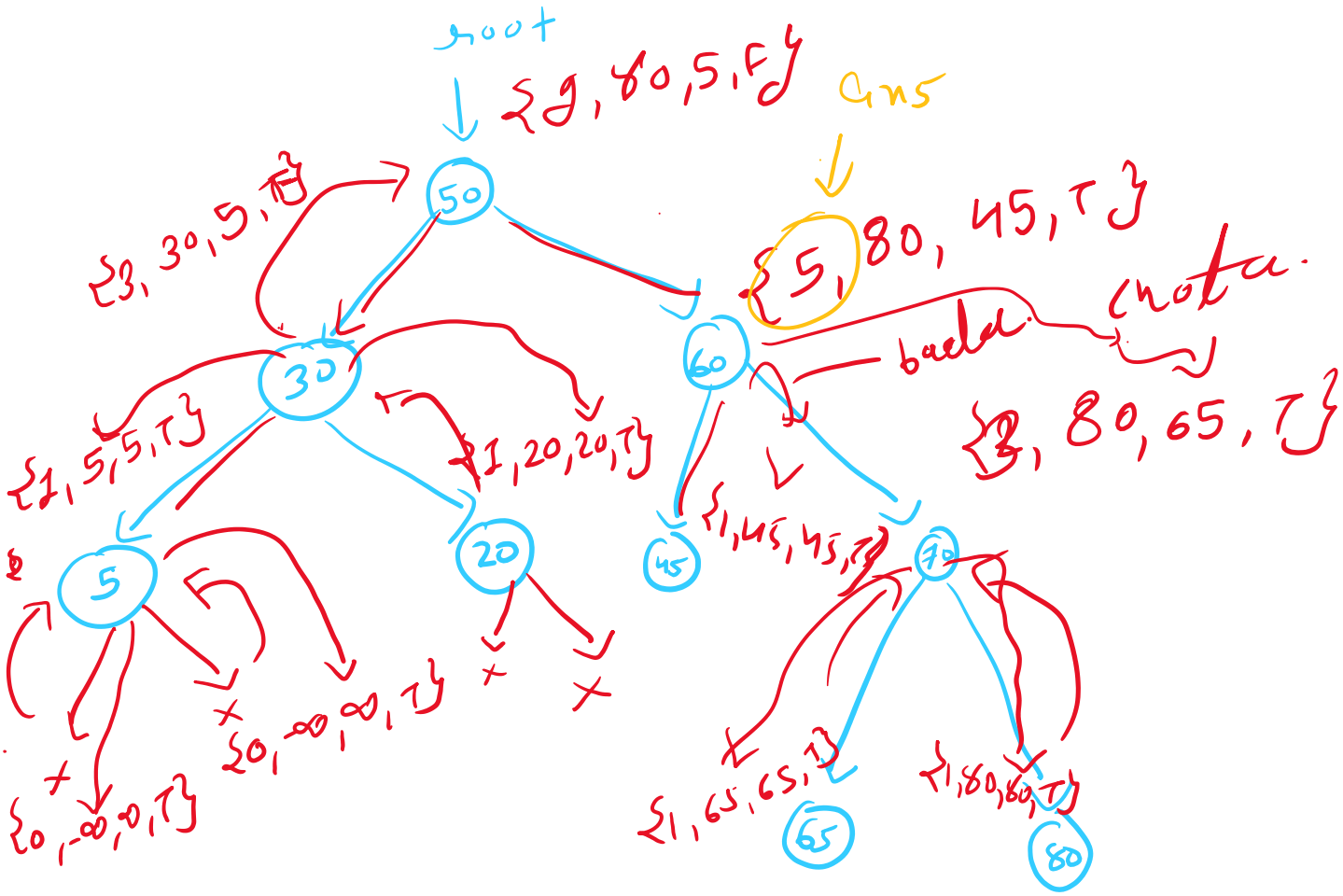




Node -> return
1) valid BST
Max Val = left
MinVal = right
Size



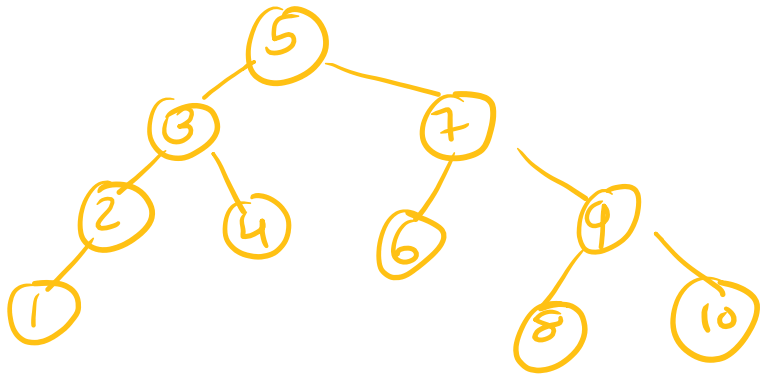



root
{9, 80, 5, F}
ans
50
30
60
5
20
45
70
65
80
{3, 30, 5, T}
{1, 5, 5, T}
{1, 20, 20, T}
{5, 80, 45, T}
{1, 45, 45, T}
{1, 65, 65, T}
{1, 80, 60, T}
{3, 80, 65, T}
backtrack
{0, -∞, ∞, T}
{0, -∞, ∞, T}

$TC = O(n)$

$SC = O(H)$

Inorder Predecessor in BST



M1 ① store inorder traw in vector

② L.S & R.S

↳ Find node just smaller them

T.C = O(n) + O(log n)

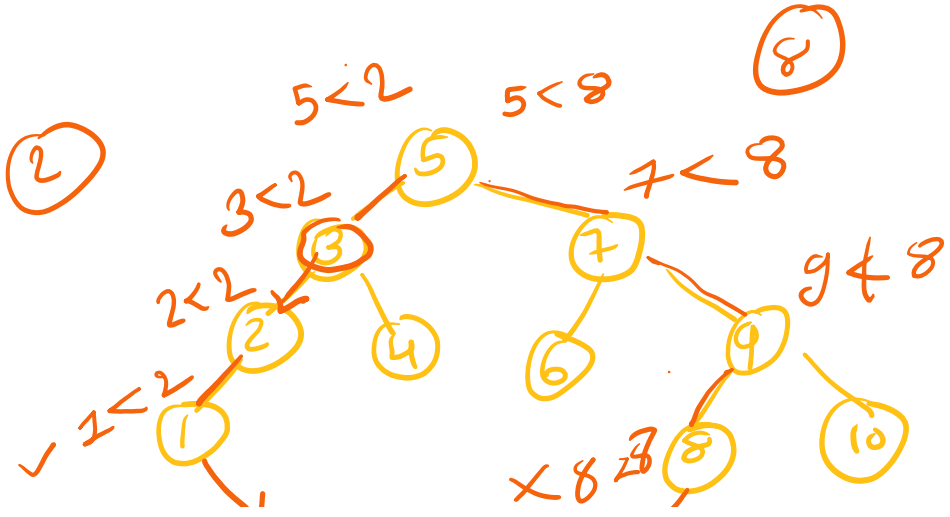
S.C = O(n)

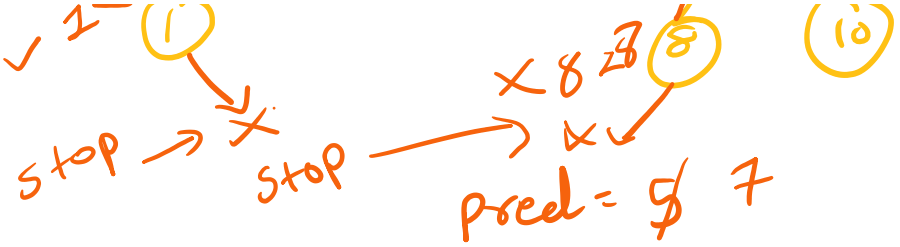
M2 perform inorder traw using RE & Find value just less them x.

T.C = O(n)

S.C = O(n)

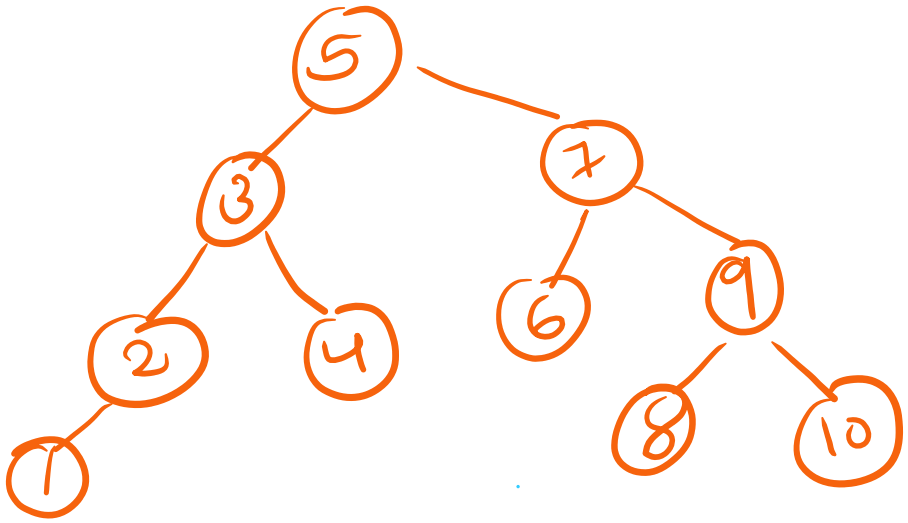
M3






```
if (root->data < target)
    root->right // prev = root->data
else
    root -> left
```

Inorder Successor in BST



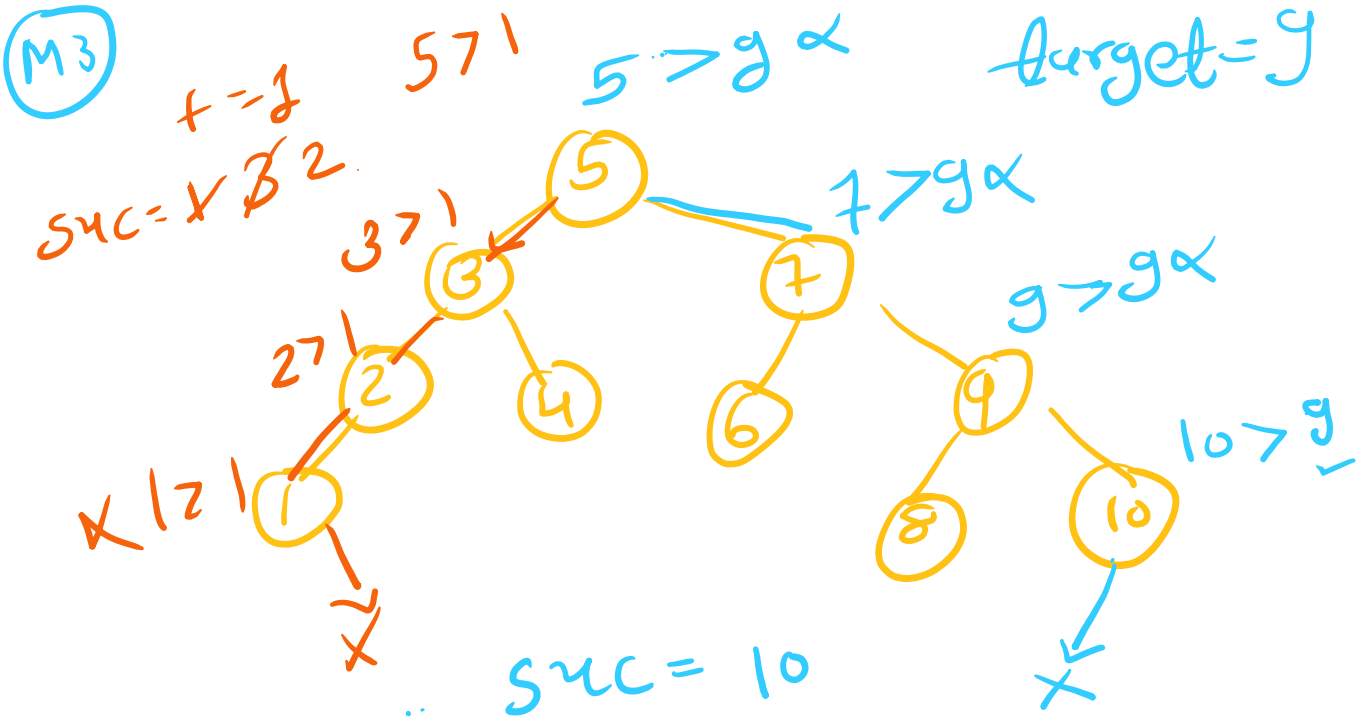
1 2 3 4 5 6 7 8 9 10

2 ka successor 3

M1 Inorder tru stor in vector

find L.S & R.S in.

M2 Inorder tru in RE and we will find value > x will be ans



TC = O(H)

SC = O(2)

(root → data > target)

suc = root → data

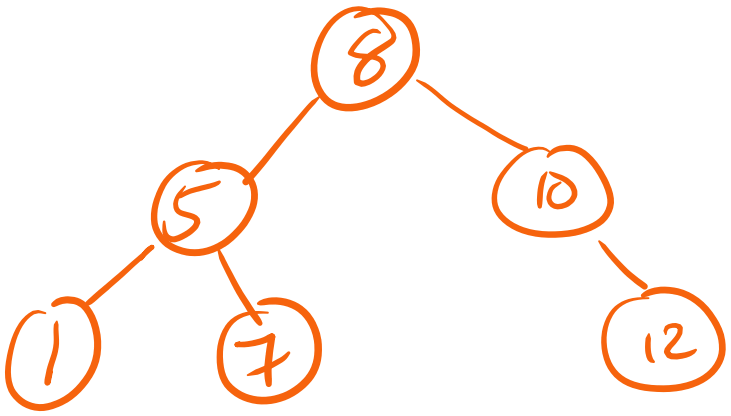
root = root → left.

else.

root = root → right

31 March 2024 11:00

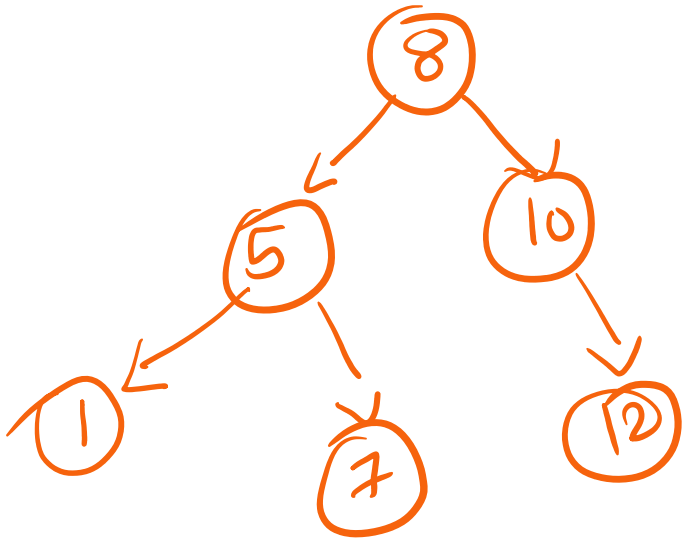
1008 Build BST using Preorder Tra



8, 5, 1, 7, 10, 12

(M1) Brute force

8 | 5 1 7 | 10 | 12



T.C = $O(n^2)$
for every node we are traversing from root.

8 5 1 7 10 12

(M2) Preorder 8 5 1 7 10 12

Inorder 1 5 7 8 10 12

sorted

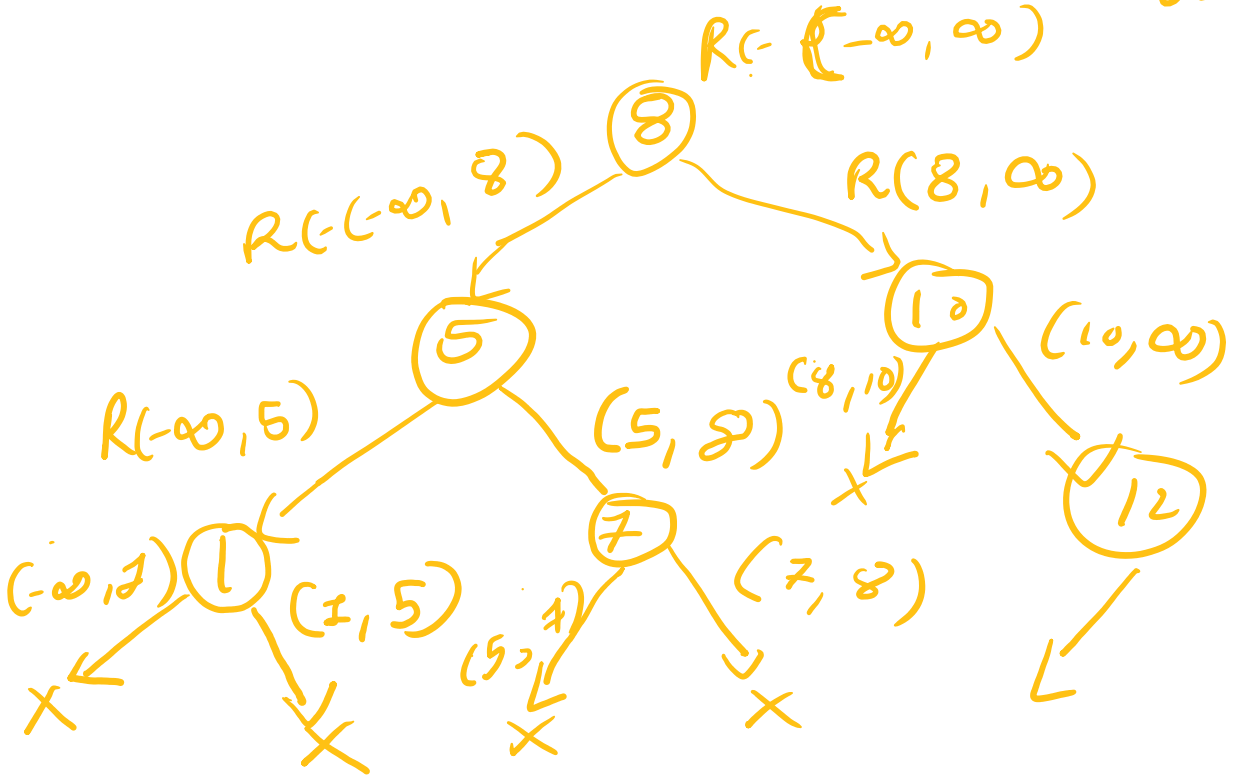
T.C = O(n log n) + O(n)

S.C = O(n)

(M3)

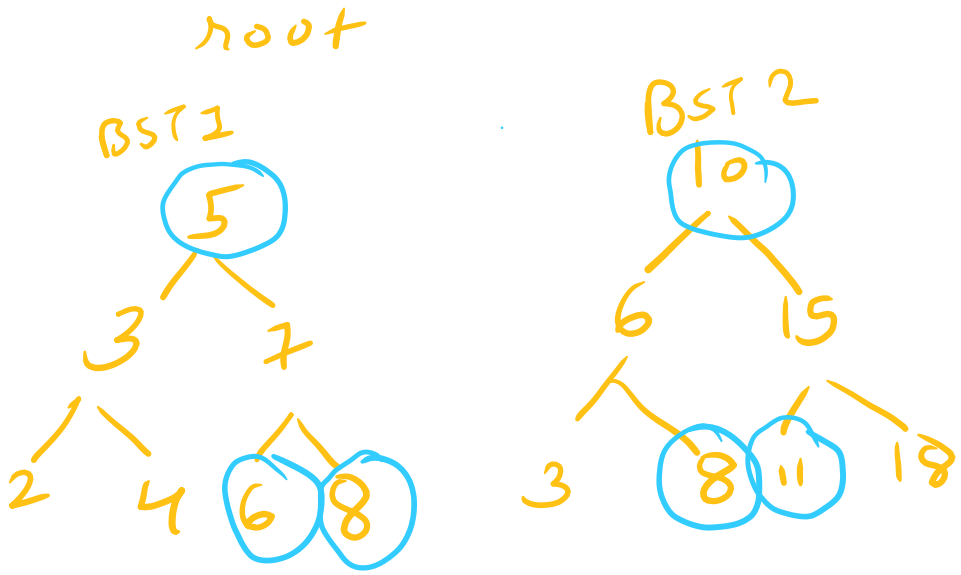
0	1	2	3	4	5
8	5	1	7	10	12
i	i	i	i	i	i

i = out of bound



T.C = O(n)

gfg Brothers from Different



X = 16 pairs of BST1 & BST2

a + b = X

(M1) Brute force $O(n^2)$

BST 1 ka 1 node lo aur BST 2 me X - x node find

(M2) BST1 (LNR) inorder

2 3 4 5 6 7 8

BST2 (RNL) inorder

18 15 11 10 8 6 3

X = 16

7 + 18 = 20 > 16

11 + 2 = 13 < 16

2+18 = 20 > 16

2+15 = 17 > 16

11+2 = 13 < 16

11+3 = 14 < 16

if SUM < X

BST1 pointer moves to its successor

if ans Find then both pointer Age badavo else

BST2 pointer moves to predecessor

11+4 = 15 < 16

11+5 = 16 = 16 ✓

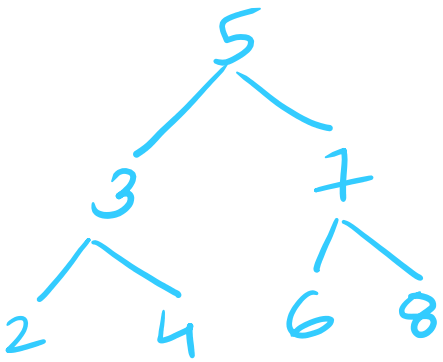
6+16 = 16 = 16 ✓

7+8 = 15 < 16

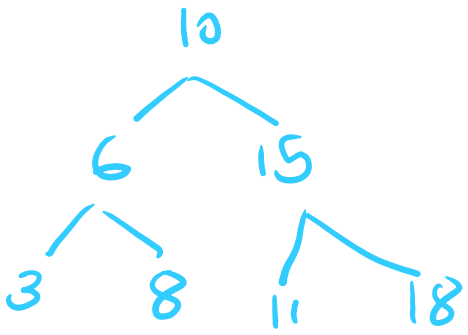
8+8 = 16 = 16

if any of out of bound then ruk jav

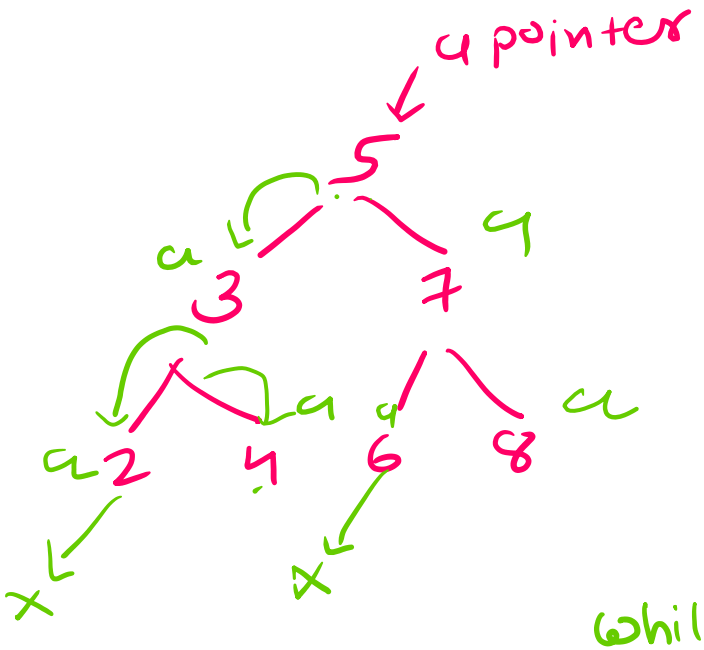
(M3) stuck → inorder tru

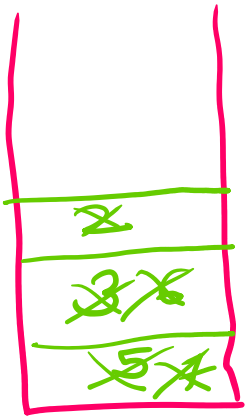


inorder tree



Reverse inorder tree





while (u)

2 3 4 5 6 7 8 &

stack. push(u)

u = u -> left

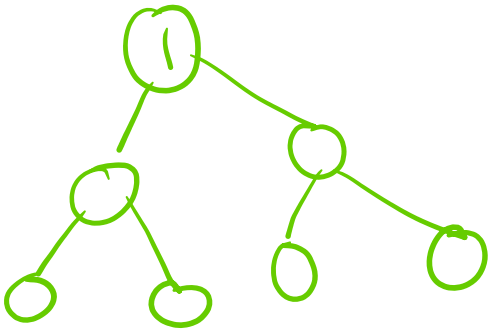
3

Reverse inorder u = u -> right.

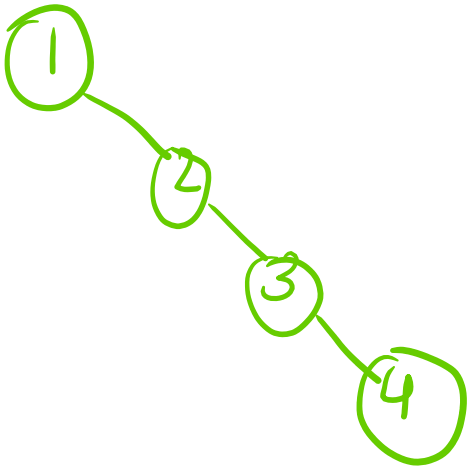
traversal =

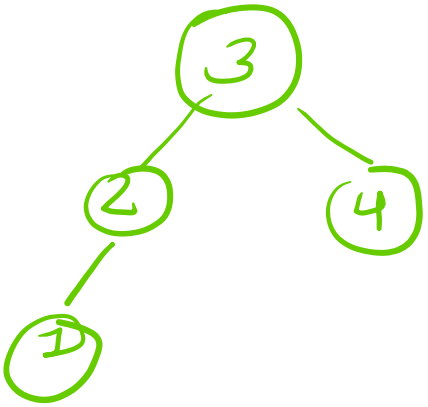
31 March 2024 17:25

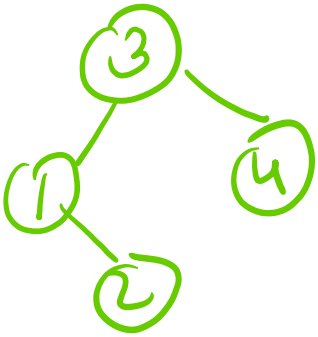
1382 Convert BST to a Balanced BST

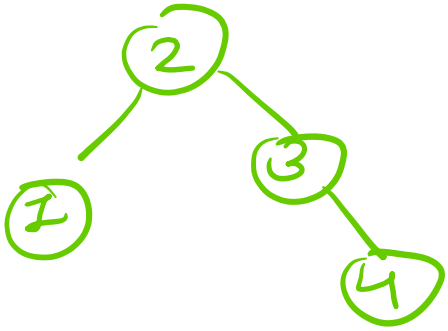


⇒ |left height − right height| <= 1





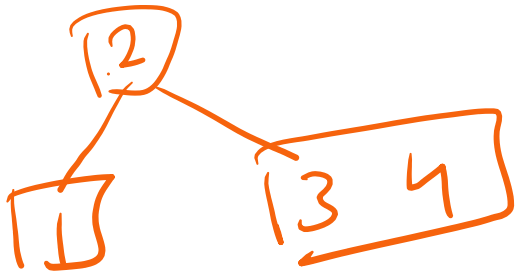




(M) inorder traw

1 2 3 4

1 2 3 4



1 2 3 4 5 6 7 8 9 10

31 March 2024 17:44

gfg

Find median OF BST

median ↑ middal ele

1 2 3 4 5 = 3 = $\frac{n+1}{2}$ odd

1 2 3 4 5 6

even = $\frac{\frac{n}{2}^{th} + \frac{n}{2}+1^{th}}{2}$

n=6 = $\frac{3+4}{2} = \frac{7}{2} = 3$

(1) Make vector using inorder

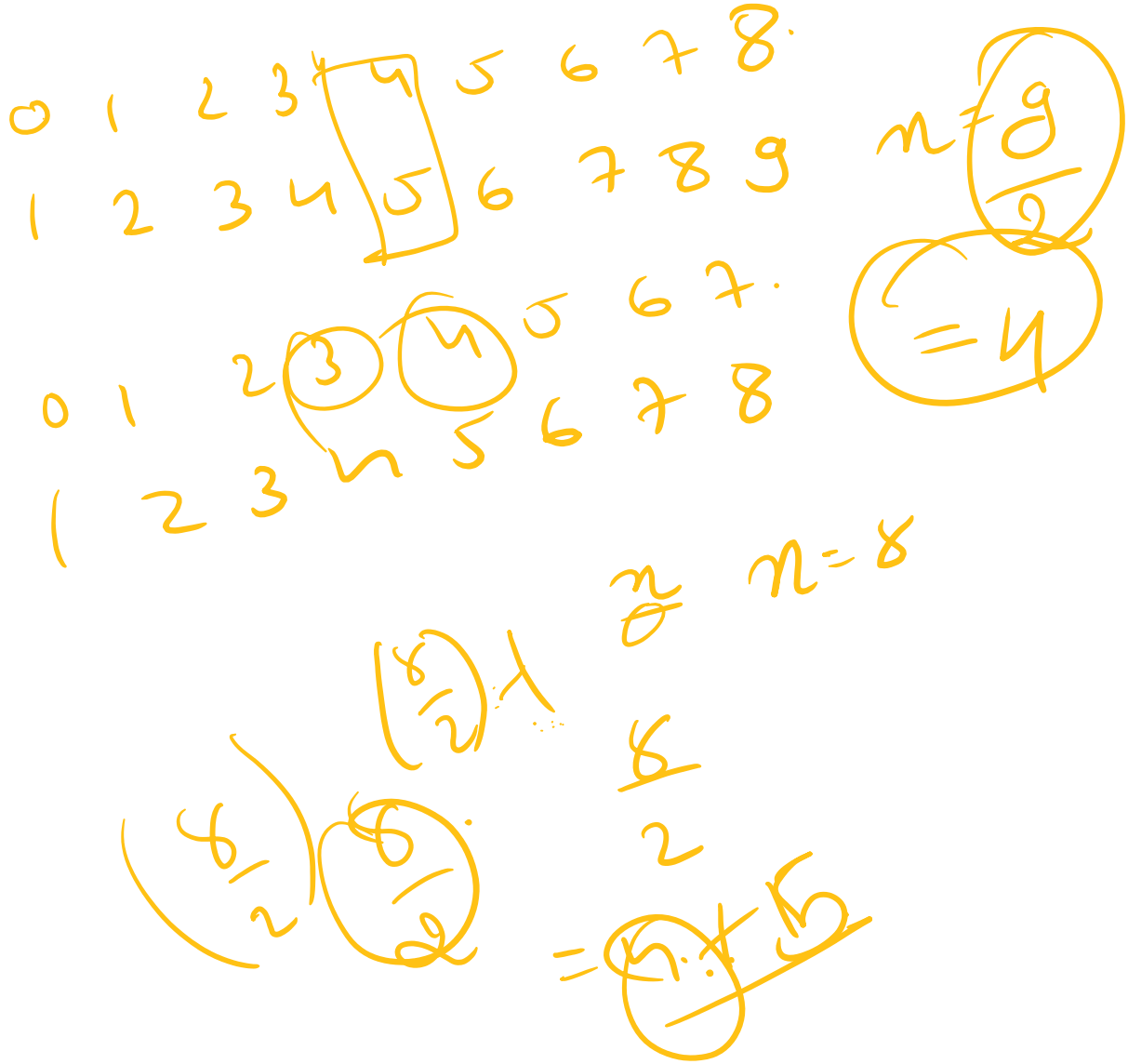
(2) Find median using formula

→ (1) RE - O(n) SC = O(n)

(2) Morris Trav TC = O(n) SC = O(1)

(1) Morris trav → no. of nodes

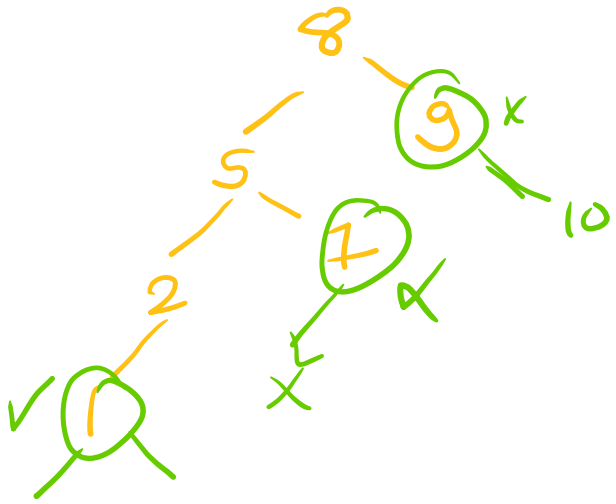
(2) Morris trav → desired element

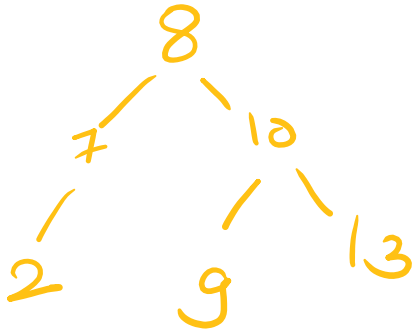


03 April 2024 16:16

gfg

Check BST has Dead End





કોઈ પણ node ની left and right if not add any node that called Dead End.

→ only check leaf node

→ any node X → if already

I am at DE

X+1 exists

X-1 exists.

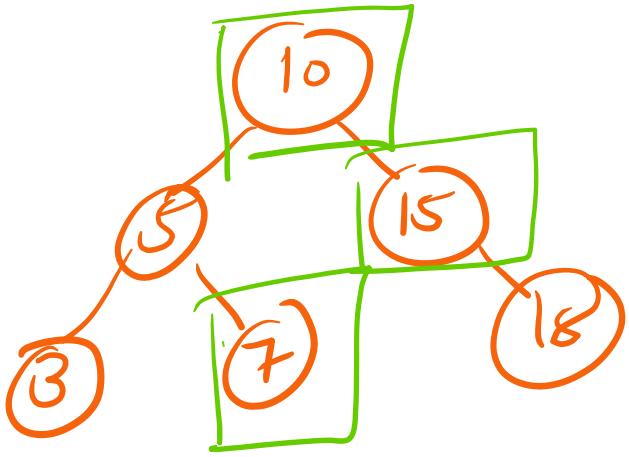
(M) → check for leaf node only

→ Pre order tru NLR

→ Pre order tra

03 April 2024 17:55

g38 Count BST Nodes lying in a Range

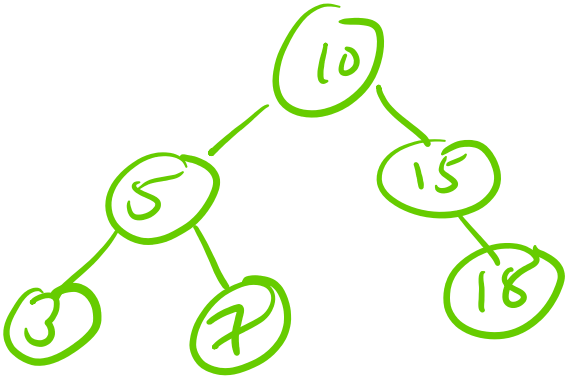


low = 7
high = 15
[7, 15]

Ans = 7 + 10 + 15 = 32

M1 -> iterative way w.s = O(n)
avg = O(n)

M2



case1 : Root value in range
-> add that node
& call left & right

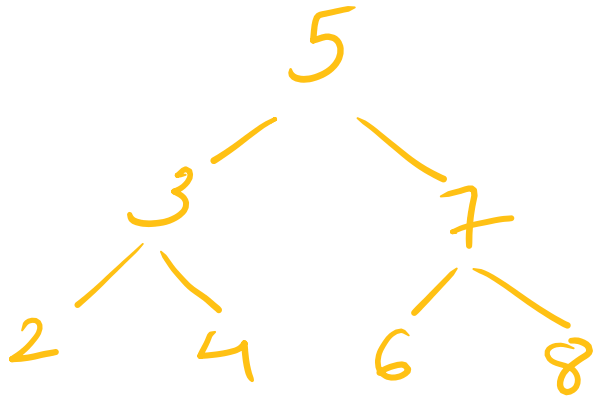
case2 :- val < low
RE -> right subtree

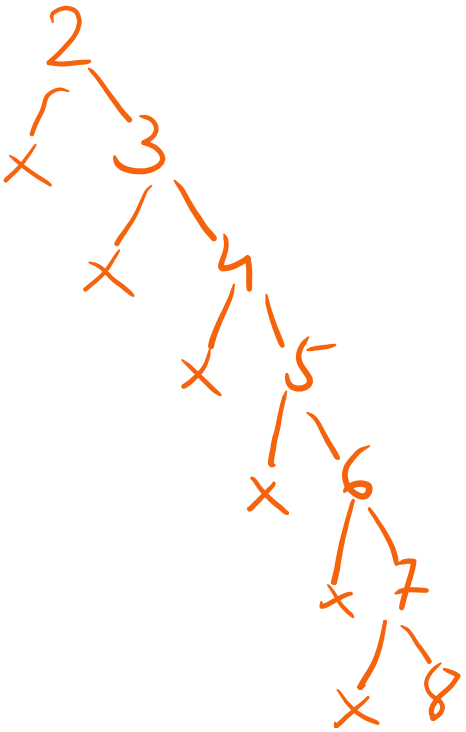
RE -> right subtree

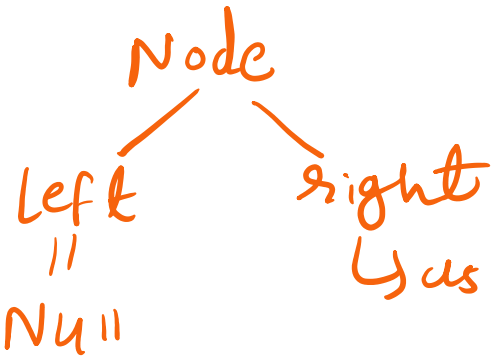
case3 :- val > high

RE :- left subtree

Flatten BST to sorted LL







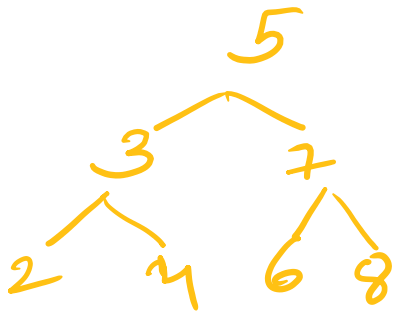
next ptr in L.L.

(M1) Recer BST from inorder

2|3|4|5|6|7|8 T.C = O(n)
S.C = O(n)

vector<int>

(M2) inorder in Node* vector & create the list



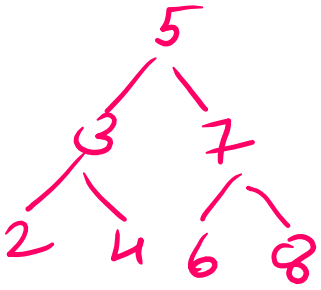
Vector<Node*>

V[i] -> left = NULL
V[i] -> right = V[i+1]

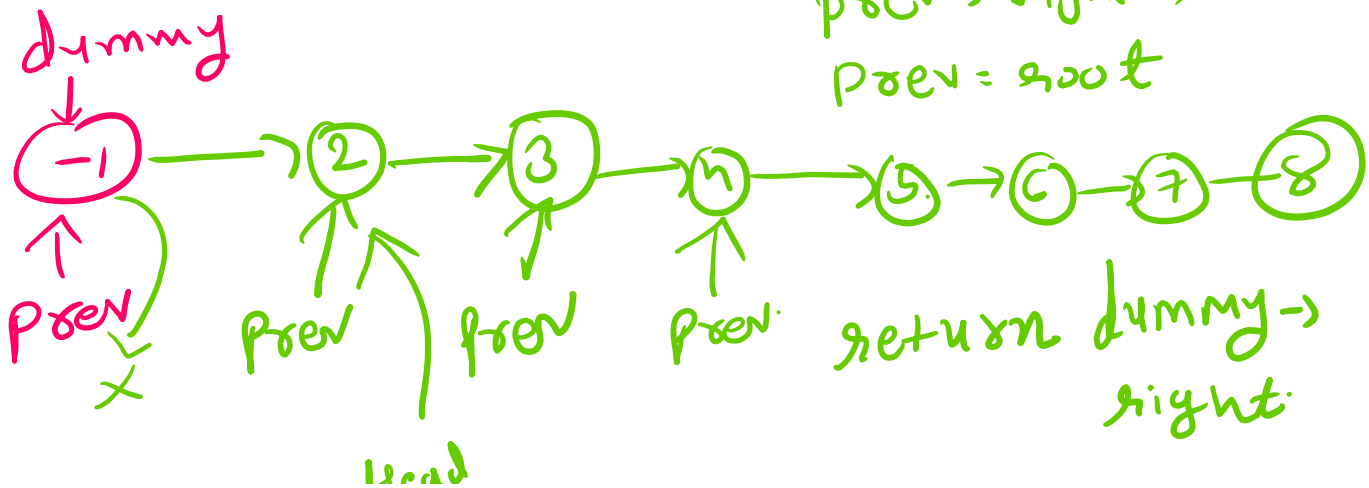
T.C = O(n)
S.C = O(n)

last element both pointer NULL

M3 Flatten inorder wise on the go.



prev->left = NULL
prev->right = root
prev = root



Replace elements with least greater element to its right.

8, 58, 71, 18, 31, 32, 63, 92, 43, 3, 91,

18 63 80 25 93

93, 25, 89, 28

-1 28 -1 -1

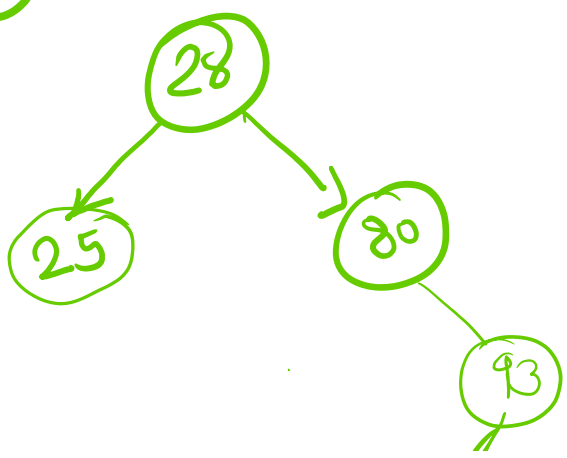
M1 for i-

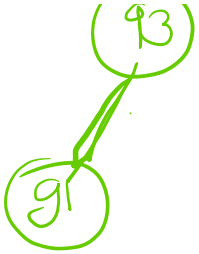
for i+1

T.C = $O(n^2)$

S.C = $O(1)$

M2 R->L array





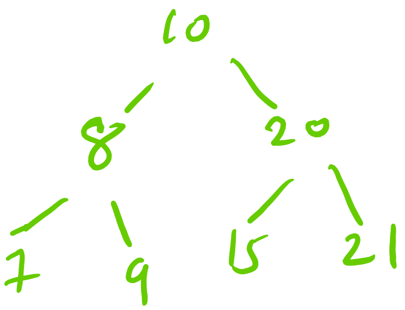
03 April 2024 19:12

Valid BST from Preorder

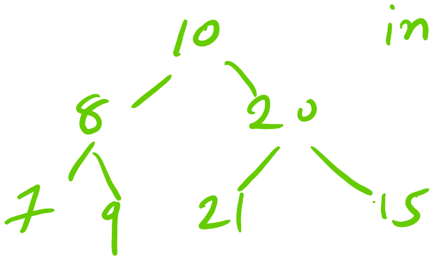
7 7 10 10 9 5 2 8

not a BST

10 8 7 9 20 15 21



valid



invalid

10 8 7 9 20 21 15

10 | 8 | 7 | 9 | 20 | 15 | 21

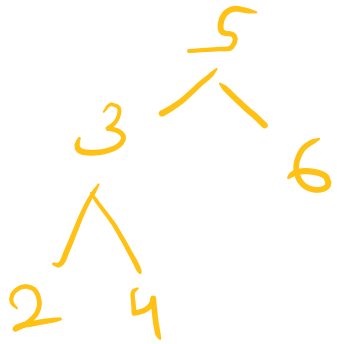
min → INT_MIN max → INT_MAX

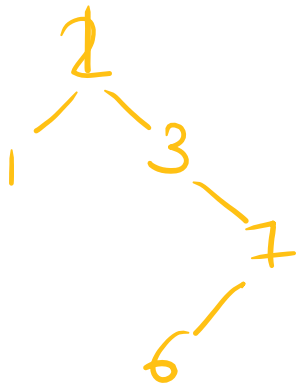
10 | 8 | 7 | 9 | 20 | 21 | 15

↑

i - અહીં આવીને stop થઈ જાય

gfg Merge two BST





(M1) BST 1 → sorted arr
BST 2 → sorted arr
TC = O(n+m)
S.C = O(n+m)

(M2) Inorder both using Stack &
keeping two pointer & merge &
make ans vector