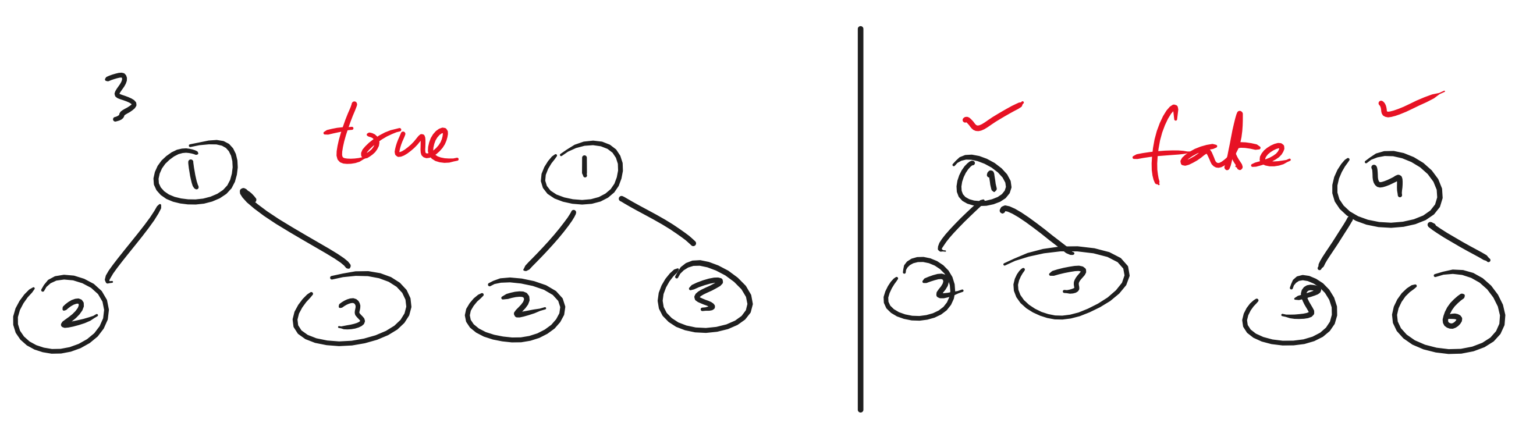
Trees continued....

* Identical Trees : →

boolean areIdentical ( $t_1$ , $t_2$ )
{



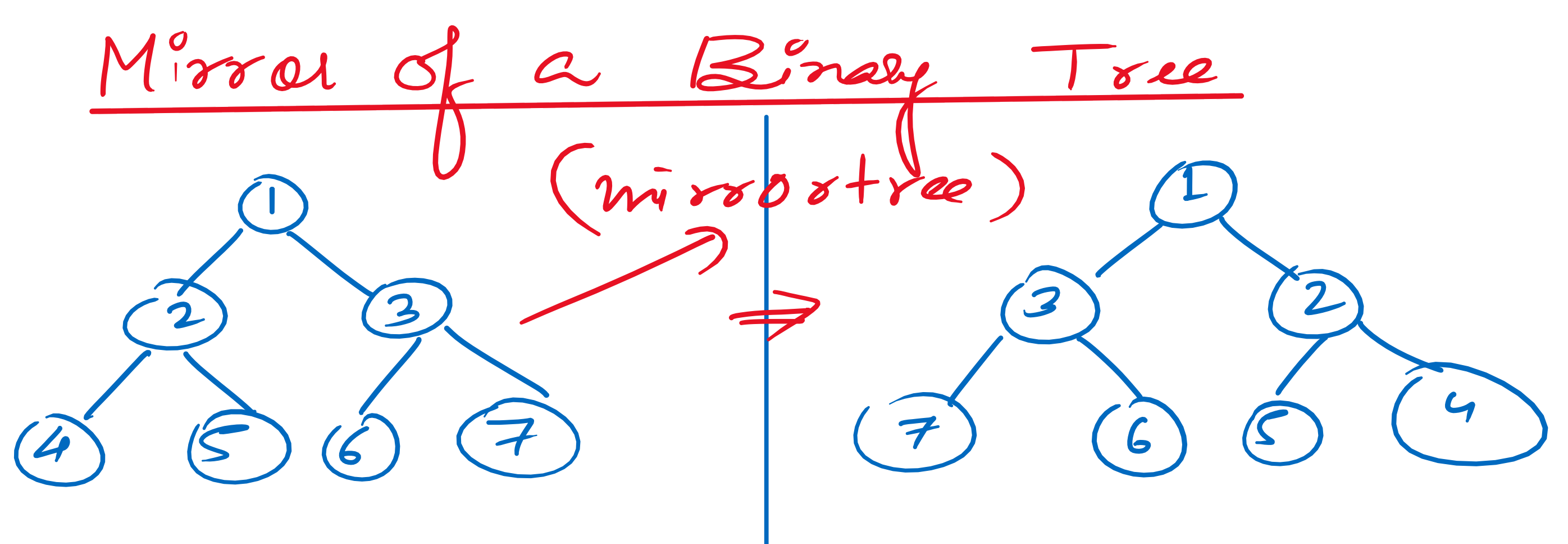
Conditional checks :

1. Both the trees are empty ⇒ true
2. One of them is empty ⇒ false
3. The data of root nodes not same ⇒ false

→ recursion

## Mirror of a Binary Tree

(mirror tree)



int temp = a ;
a = b ;
b = temp ;

Inorder : 4 2 5 1 6 3 7
Recursion

temp = r.r ;
r.r = r.l ;
r.l = temp ;
(Nodes)
In order :
7 3 6 1 5 2 4

## Introduction to Binary Search Tree :

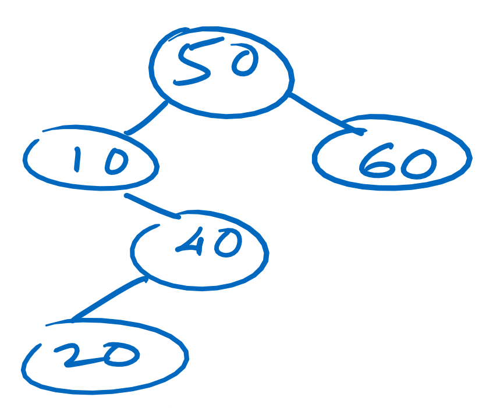
The BST follows a very important property for each Node :

Each Node 'N' ⇒ L < N < R

So, the node that's less than root goes to the left & node that's greater than the root goes to the right.

int[] arr = { 50, 10, 60, 40, 20 }

if 50 is assumed to be root :



So, each node : L < N < R

⑨⓪ > root ⇒ left side is skipped

⑤ < root → right side is skipped

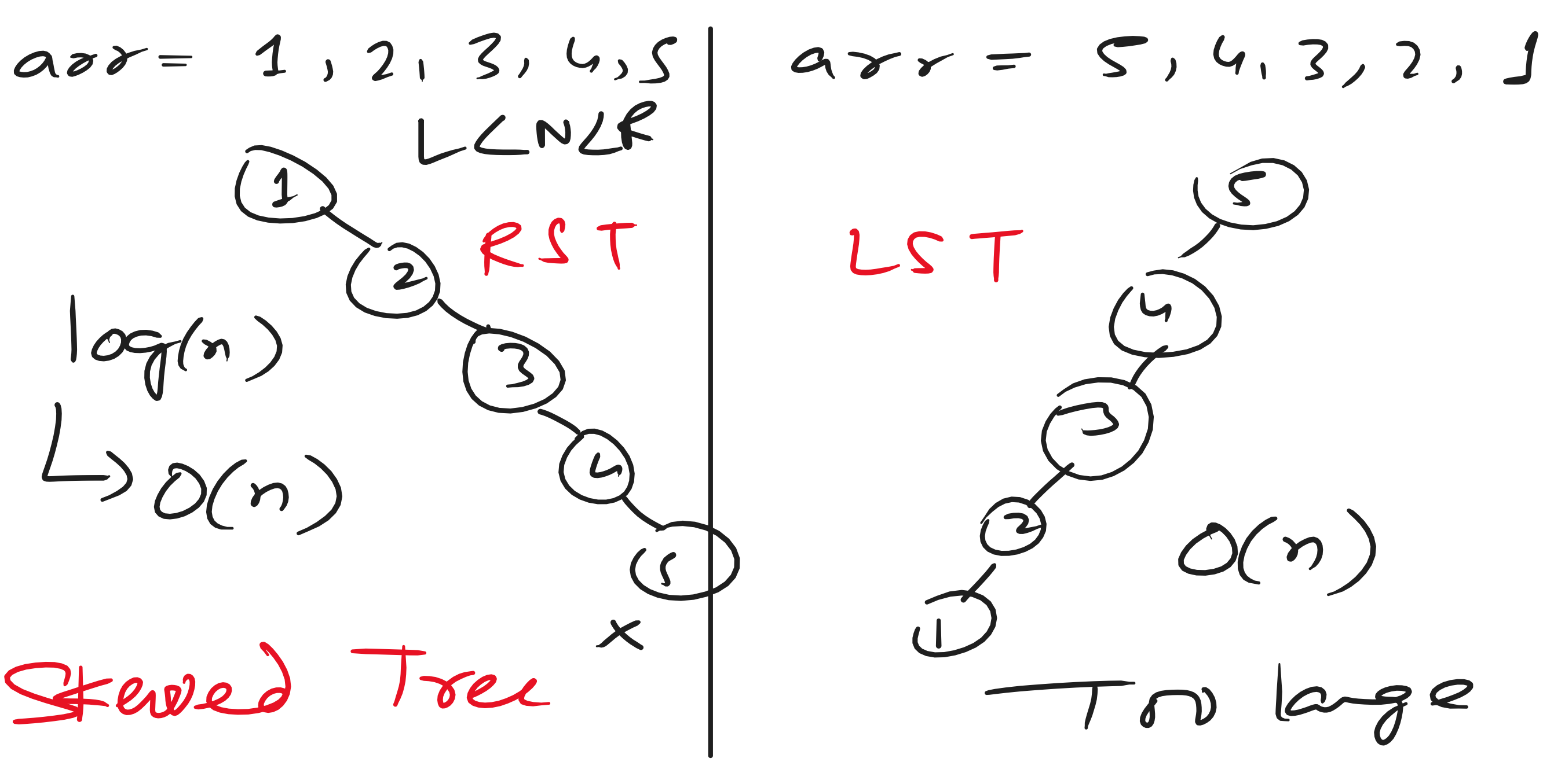
Big O (log n)

What is the disadvantage / drawback of BST ?

arr = 1, 2, 3, 4, 5

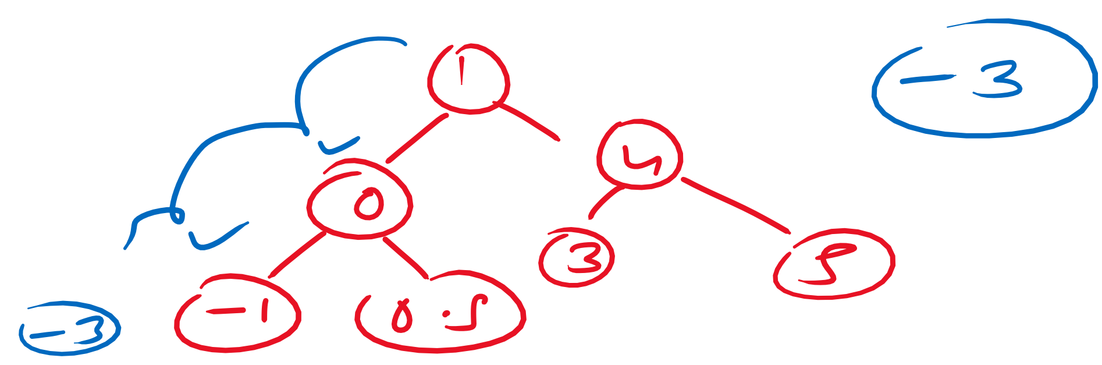
L<N<R

arr = 5, 4, 3, 2, 1



* Implementation & methods :

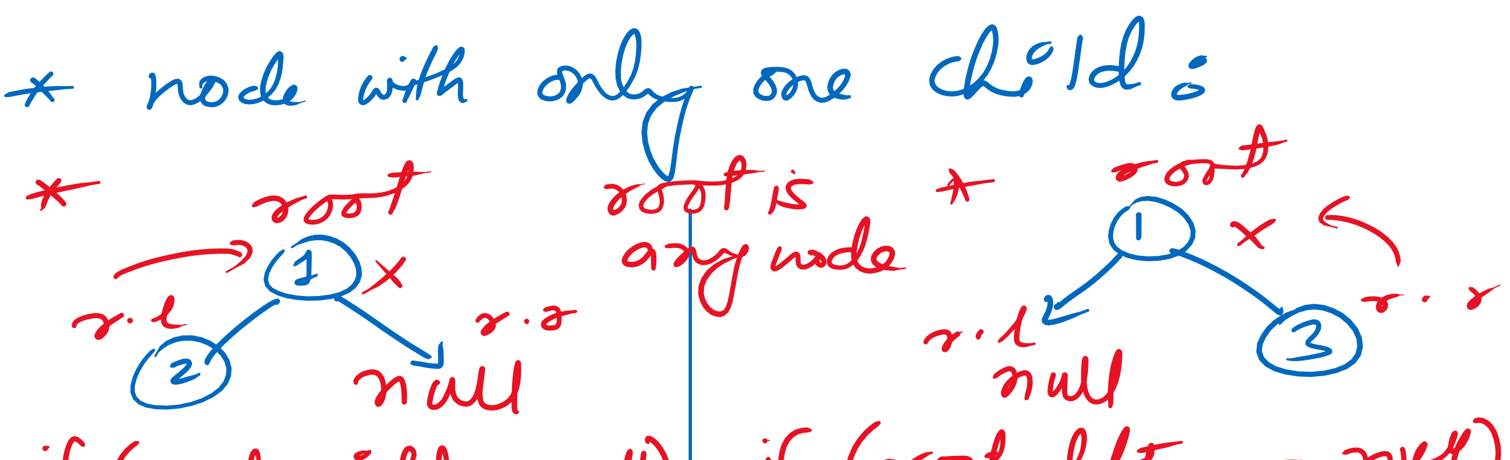
- * insert → L<N<R
- * search → L<N<R
- * delete → L<N<R



In-order Ascending order

Delete method important factors to consider :

* node with only one child :

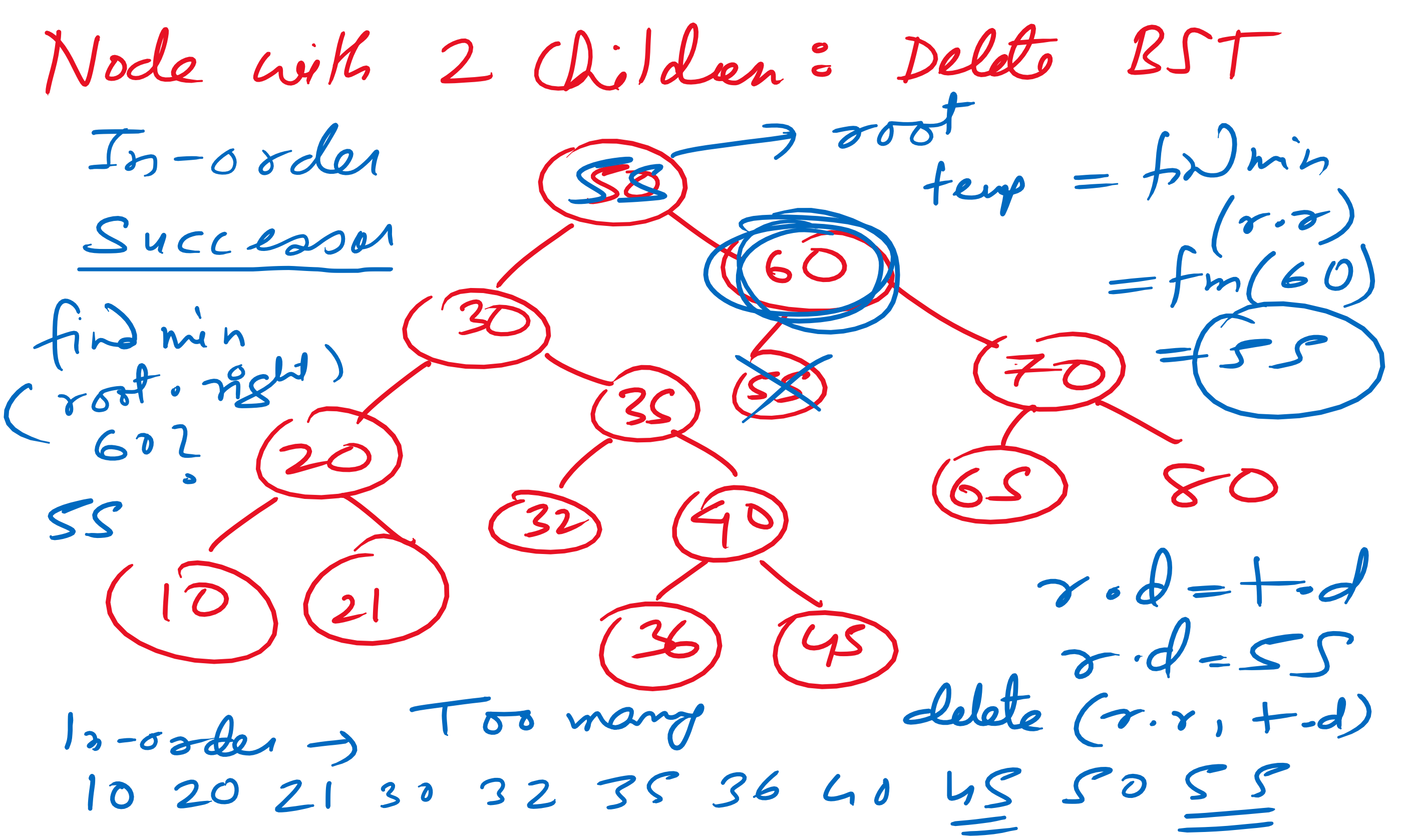


if (root.right == null)
Node temp = r.l
~~delete~~ root = null
C++ return temp
JVM

if (root.left == null)
Node temp = r.r
~~delete~~ root = null
C++ return temp

Node with 2 Children : Delete BST

In-order
Successor



temp = findmin (r.r)
= fm(60)
= 55

r.d = t.d
r.d = 55
delete (r.r, t.d)

In-order → Too many
10 20 21 30 32 35 36 40 45 50 55
60 65 70

## Sorted Array to Balanced BST :

Normally, when we take a sorted array, the BST formed is skewed (right or left) & so the time complexity of all operations becomes : → O(n) from log(n).

How can you convert a sorted array into Balanced BST ?

$$\begin{array}{c} s \\ 0 \end{array}\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ \begin{array}{c} e \\ 8 \end{array}$$

arr = { 1, 2, 3, 4, 5, 6, 7, 8, 9 } ;

Find the mid $\frac{s+e}{2} = \frac{8}{2} = 4$

root = arr[mid] = ⑤

recursion

0, ±1