DSA through C++

AVL tree

Saurabh Shukla (MySirG)

Agenda

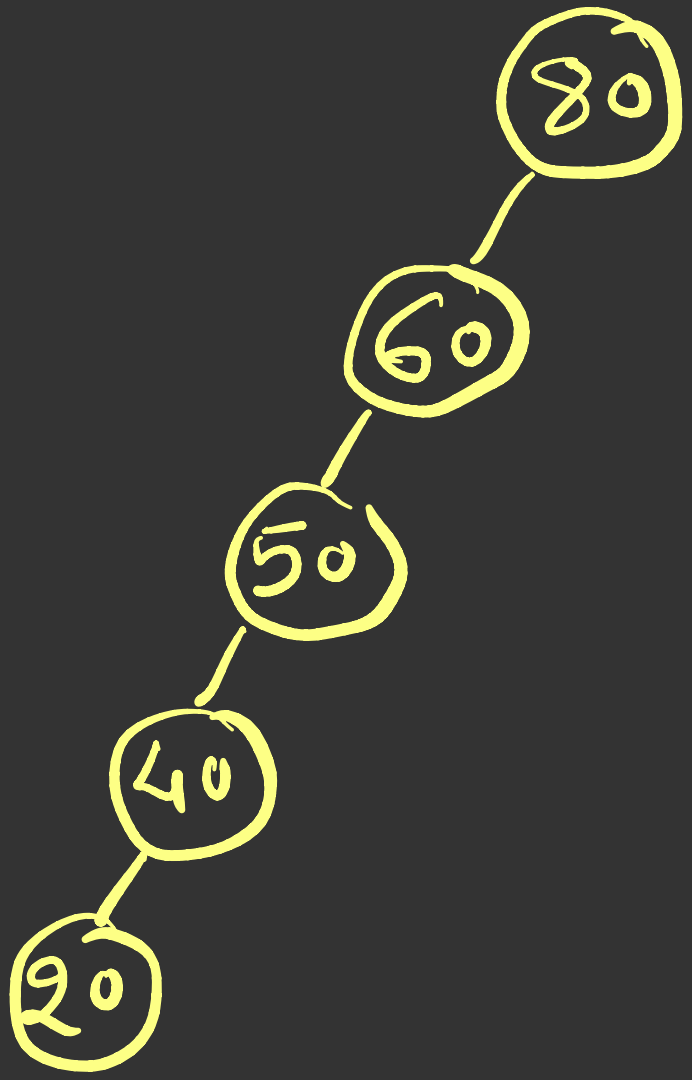
1. Problem with BST
2. Balanced Height
3. AVL tree
4. Balance factor
5. Structure
6. Rotations
7. Implementation

Problem with BST

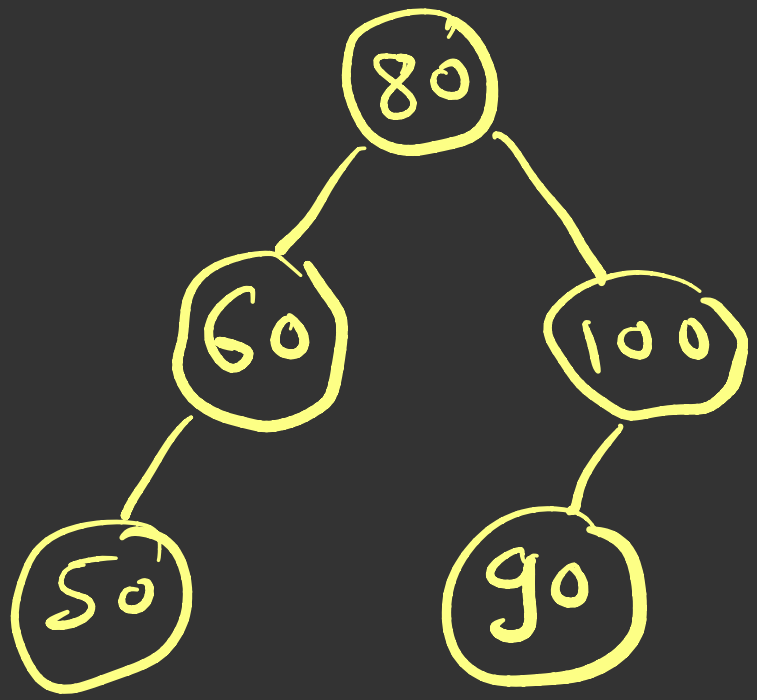
The average time complexity of Searching an element in BST is $O(\log_2 n)$

Searching time increases if the BST is skewed.

80, 60, 50, 40, 20



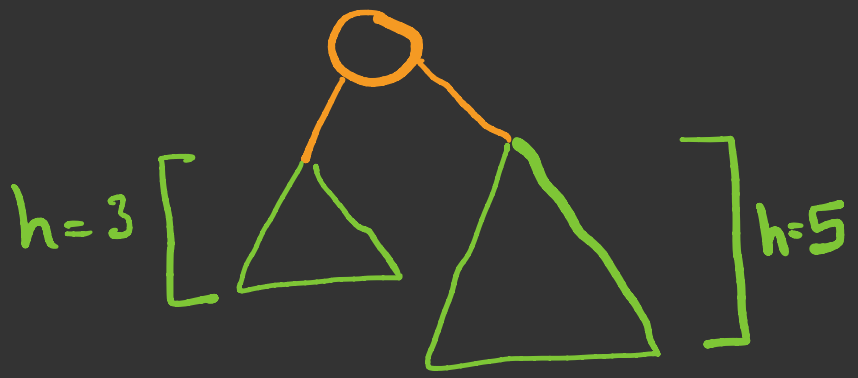
80 60 100 50 90

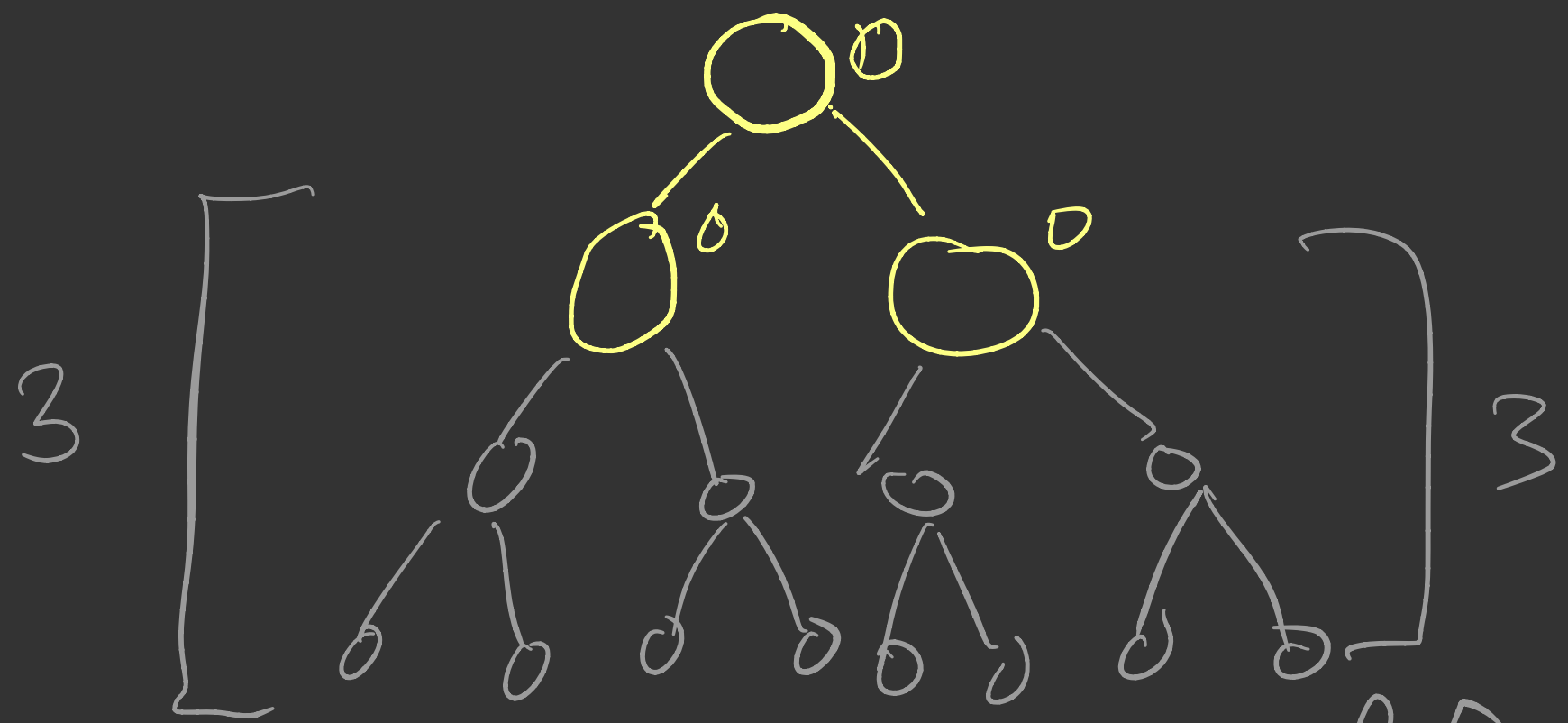


The disadvantage of a skewed binary search tree is that the worst case time complexity of a search is $O(n)$

Balanced Height

There is a need to maintain the BST to be of balanced height, so that it is possible to obtain for the search option a time complexity of $O(\log_2 n)$ in the worst case





$$BF = H(T_L) - H(T_R)$$

1

-1

AVL

One of the popular balanced tree was introduced by Adelson-Velskii and Landis (AVL)

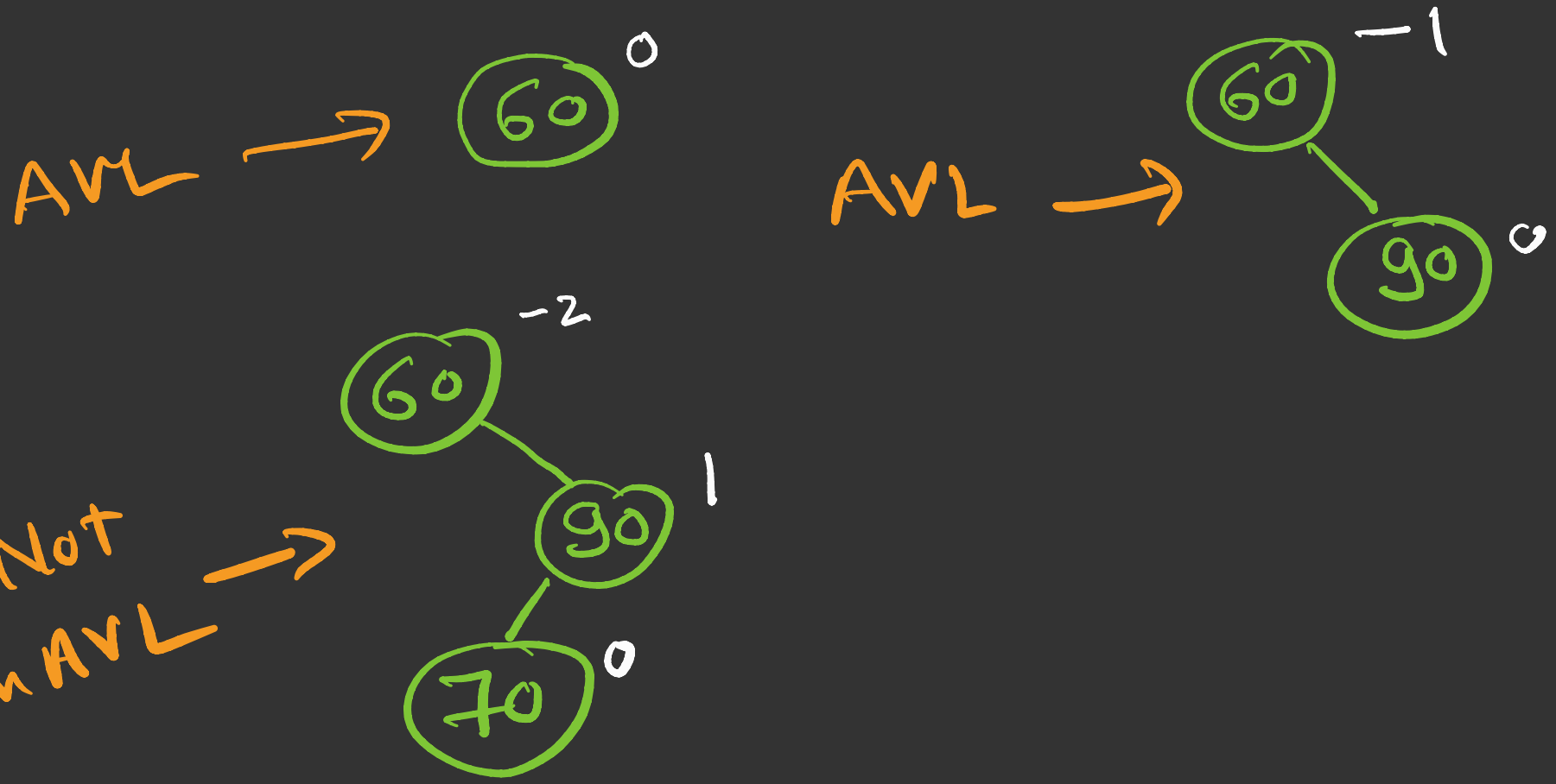
An empty binary tree is an AVL tree

A non empty binary tree T is an AVL tree iff given T_L and T_R to be the left and right subtrees of T and $h(T_L)$ and $h(T_R)$ to be the heights of subtrees T_L and T_R respectively, T_L and T_R are AVL trees and

$$|h(T_L) - h(T_R)| \leq 1$$

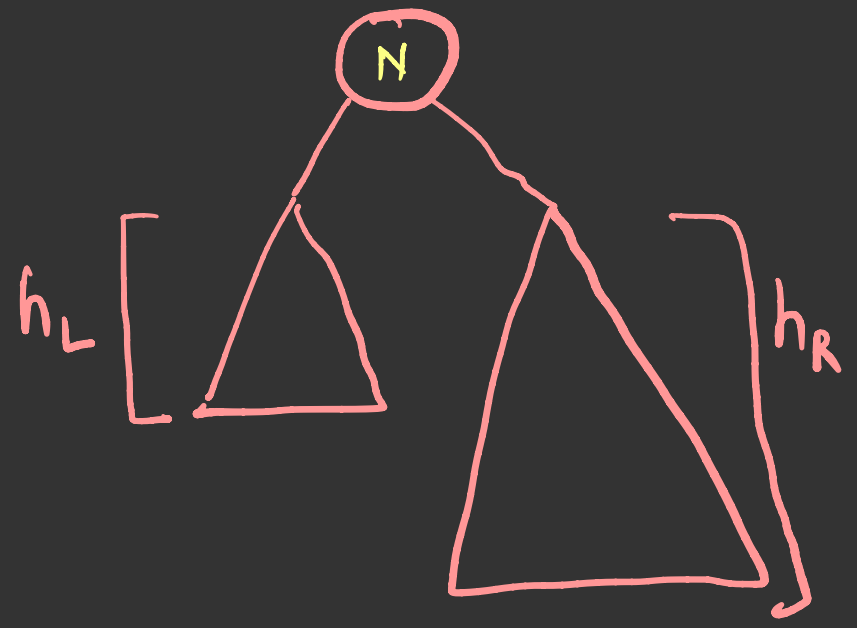
Balance Factor $= h(T_L) - h(T_R)$

For AVL tree the balance factor of a node can be either -1, 0 or 1



AVL tree = BST + BF of all the nodes must be -1, 0, or 1

```
struct node
{
        node *left;
        int item;
        node *right;
        int height;
};
```



Height of N is

$$\max(h_L, h_R) + 1$$

AVL tree is a self balancing tree.

insertion and deletion in AVL tree can disturb the balance factor of inserted or deleted node along with their anscesters.

AVL tree apply rotations to gain back AVL status by keeping balance factor within the permissible range

Rotations

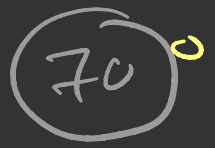
LL rotation
RR rotation
} Single Rotation

A को B का child बना देना है

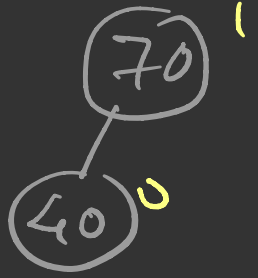
LR rotation
RL rotation
} Double Rotation

A & B को C का child बना दो

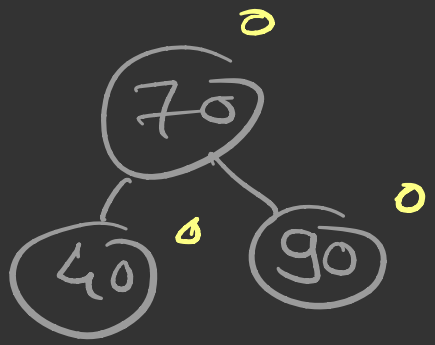
70, 40, 90, 100, 120



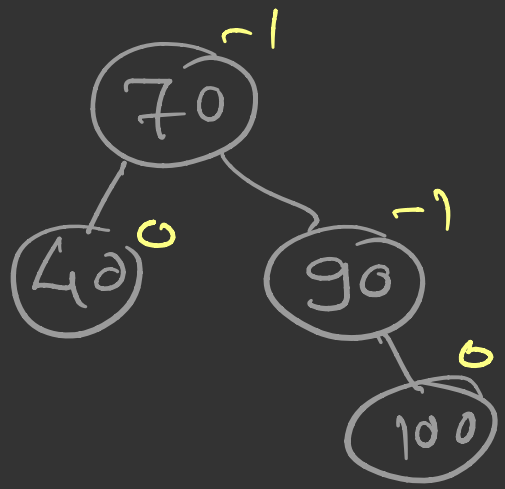
AVL



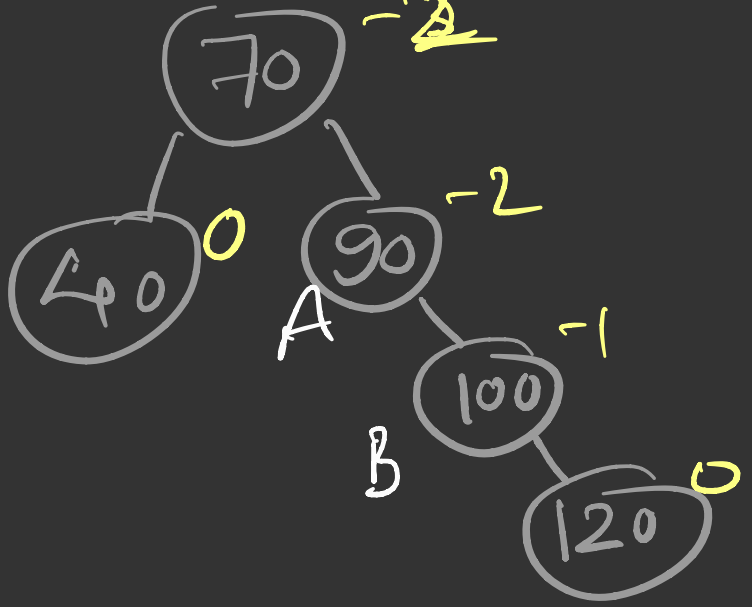
AVL



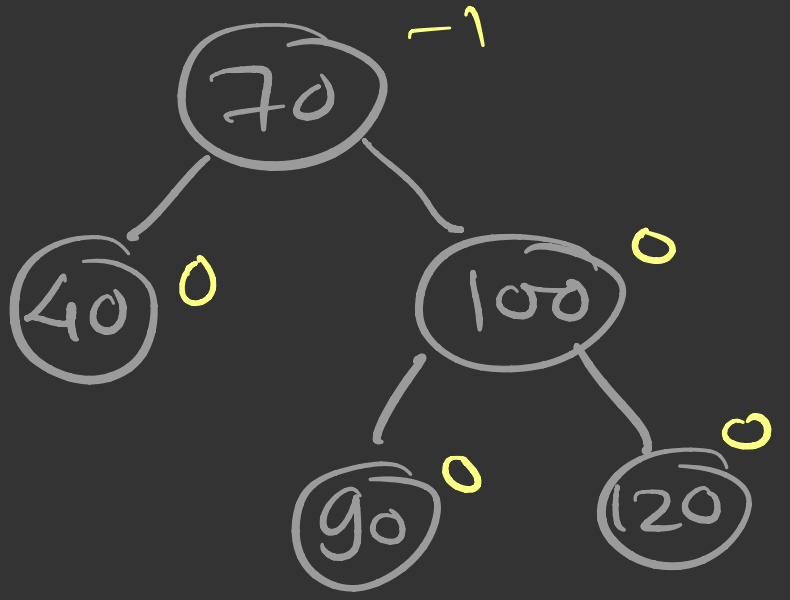
AVL



AVL

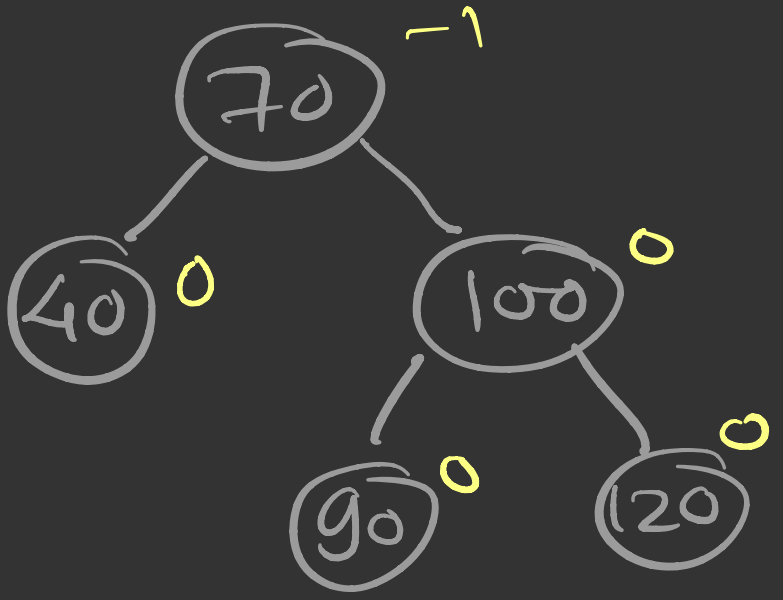


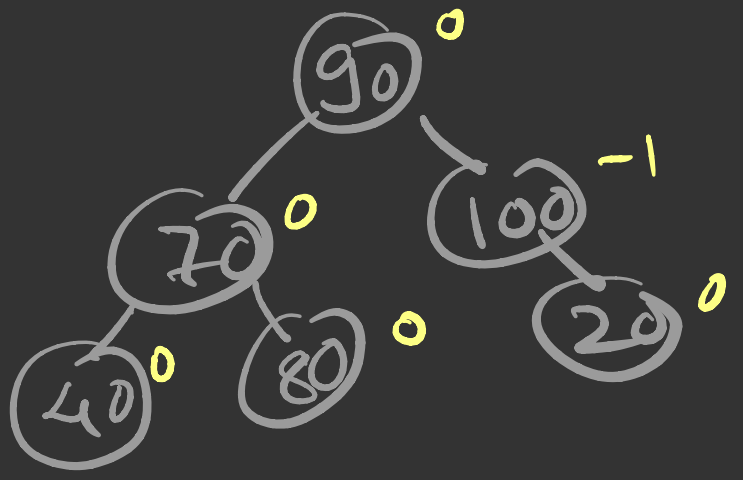
RR Rotation



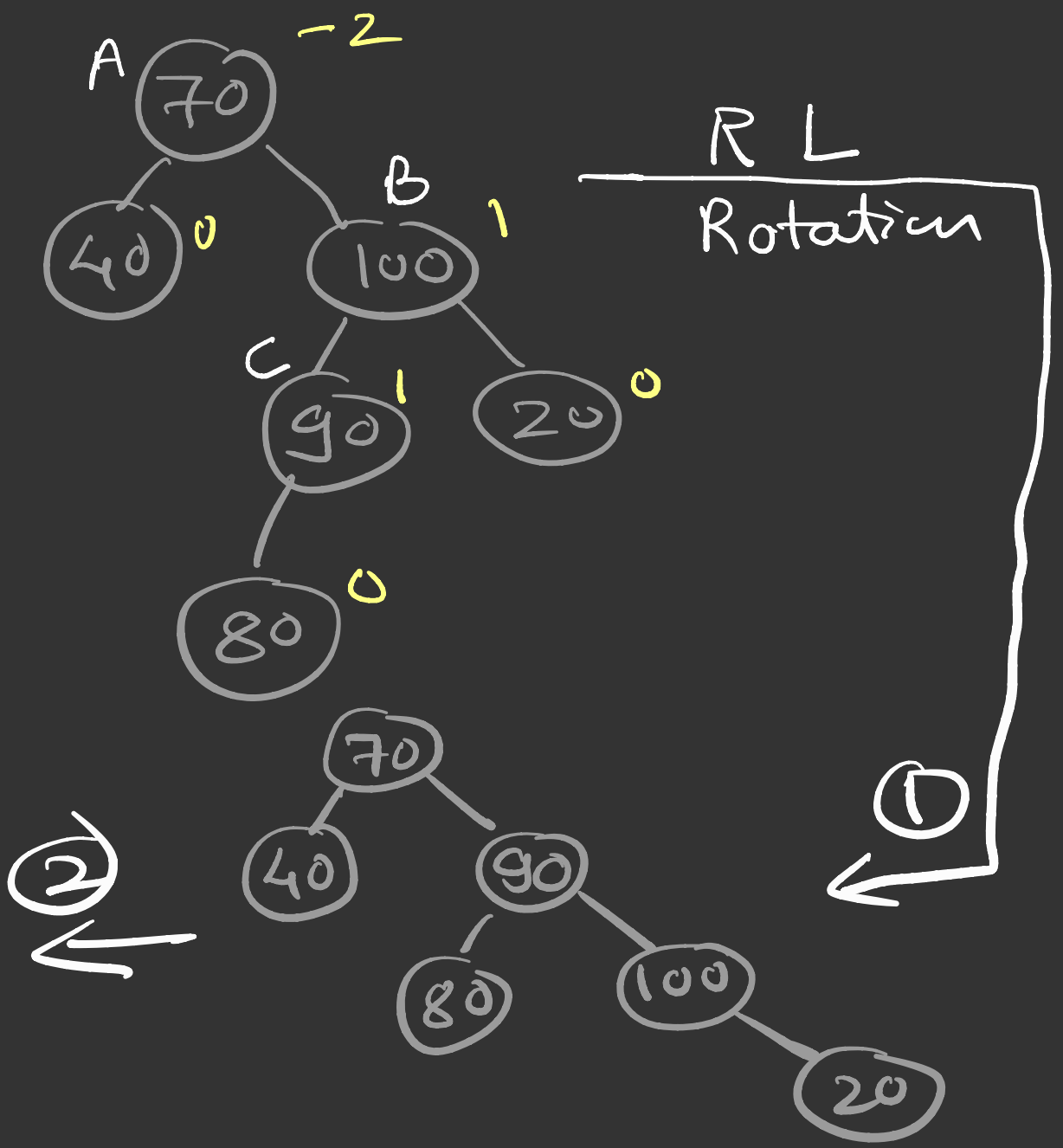
AVL

80


70
-1
40
0
100
0
90
0
120
0


90
0
70
0
100
-1
40
0
80
0
20
0

AVL


A
70
-2
40
0
B
100
1
C
90
1
20
0
80
0
R L
Rotation
1
2
70
40
90
80
100
20