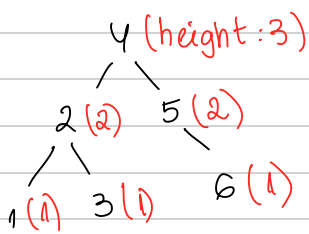
AVL Trees

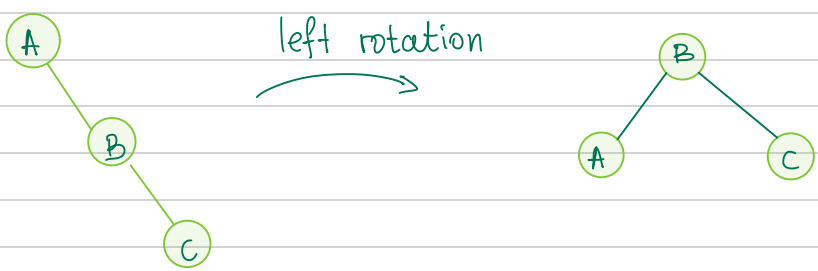
An AVL tree is defined as a self-balancing Binary Search Tree (BST) where the difference between heights of the left and the right subtrees for any node cannot be more than one

Example of AVL tree:

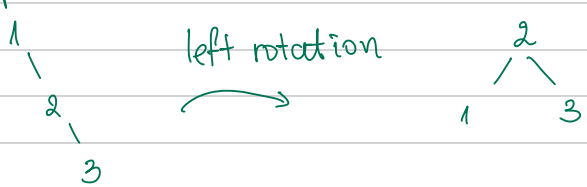


Maintaining the tree balanced:

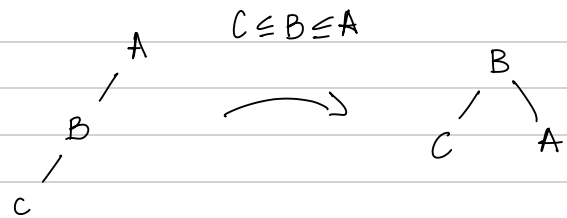
Left rotation



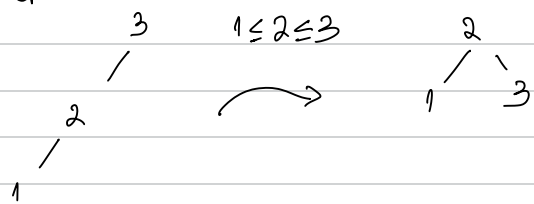
Пример:



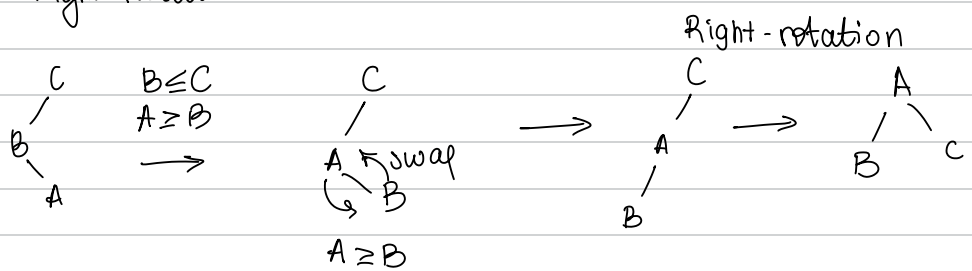
Right rotation



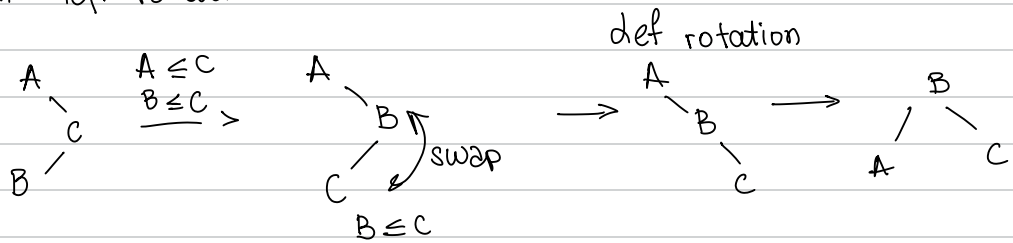
Пример:



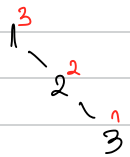
left - Right Rotation



Right left rotation



Insert: 1, 2, 3



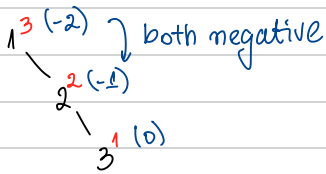
Balance factor: h_l - h_r
- h_l: height of the left subtree
- h_r: height of the right subtree

Проверяме [balance_factor = bf] bf на връх 3. (bf node)

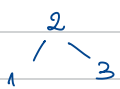
$bf_3 = 0 - 0$ (няма наследници)

$bf_2 = 0 - 1 = -1$

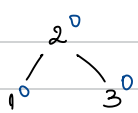
$bf_1 = 0 - 2 = -2$ ↝ need to rebalance



left rot.

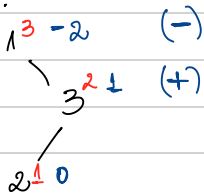


right rotation



left-right rotation

Insert 2:

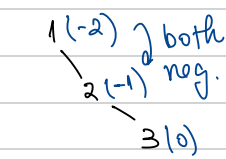


$bf_2 : 0$

$bf_3 : 1 - 0 = 1$

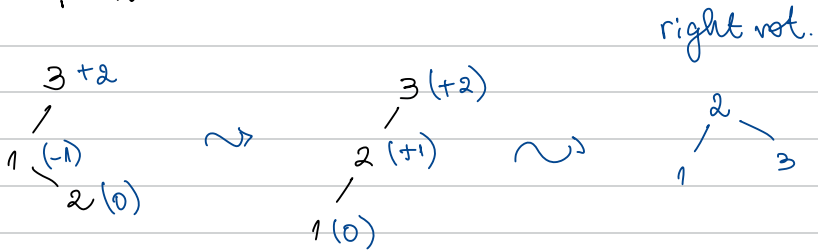
$bf_1 : 0 - 2 = -2$

making a straight line

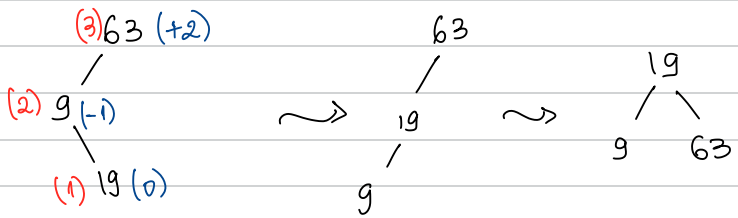


left rotation

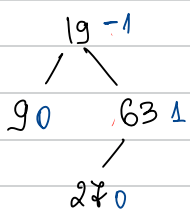
right-left rotation



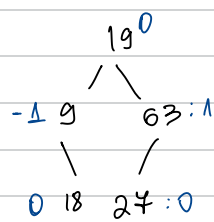
Insert 63, 9, 19, 27, 18, 108, 99, 81



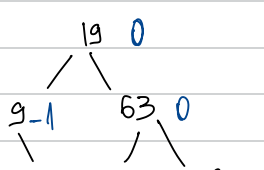
insert 27



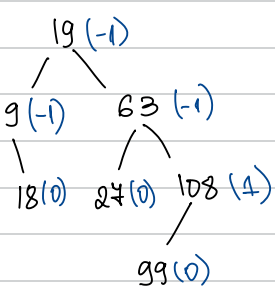
insert 18



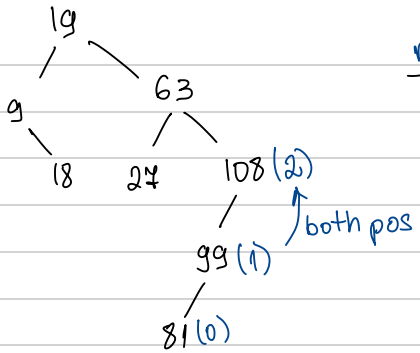
insert 108



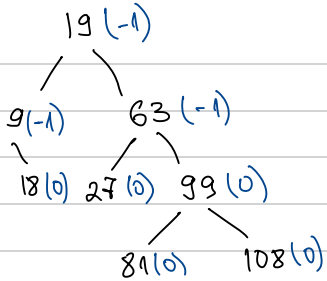
insert 99



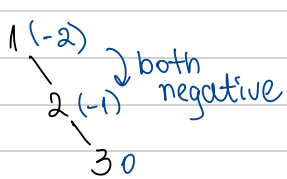
insert 81 →



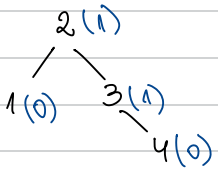
rotate →



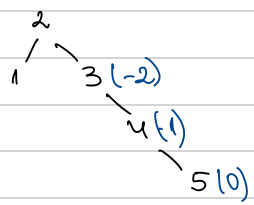
Insert : 1, 2, 3, 4, 5, 6



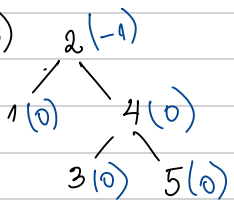
4 ⇝



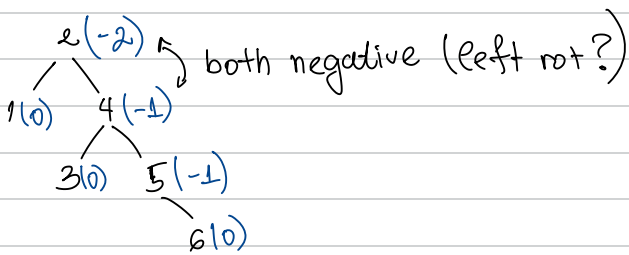
5 ⇝



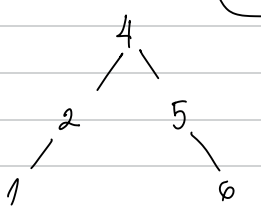
⇝ (left)



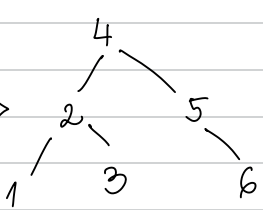
6 ⇝



forget for a moment



⇝



How do we define height: $H(n) = \max(H(T_L), T(T_R)) + 1$

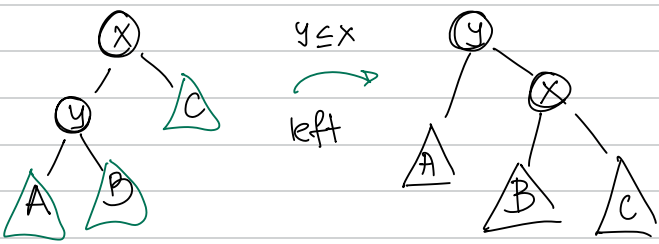
$H(\emptyset) = -1$ ↔ $H(\emptyset) = 0$

$H(\text{leaf}) = 0$ ↔ $H(\text{leaf}) = 1$

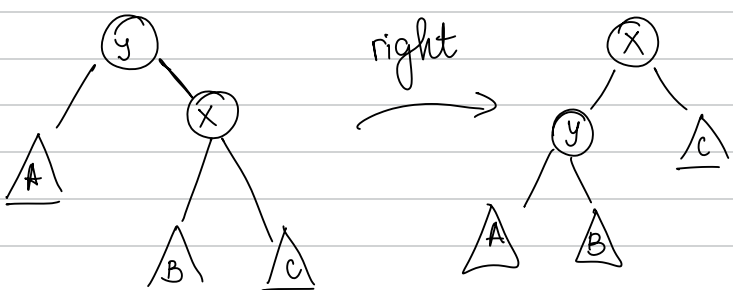
$BF(n) = H(T_L) - H(T_R)$

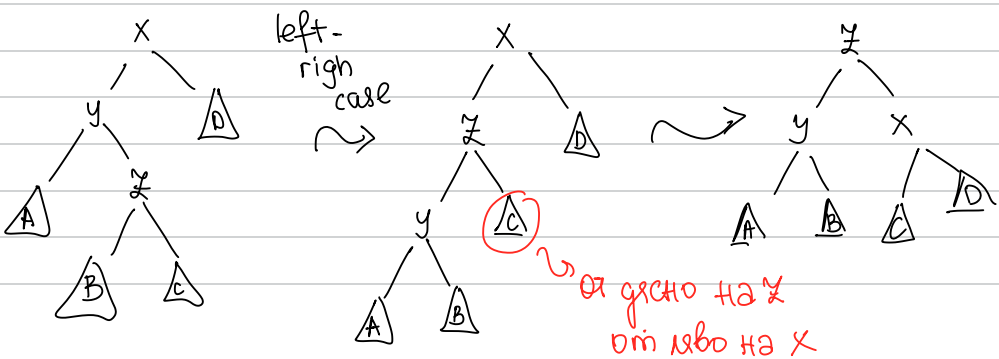
AVL Tree = $|BF(n)| \leq 1$

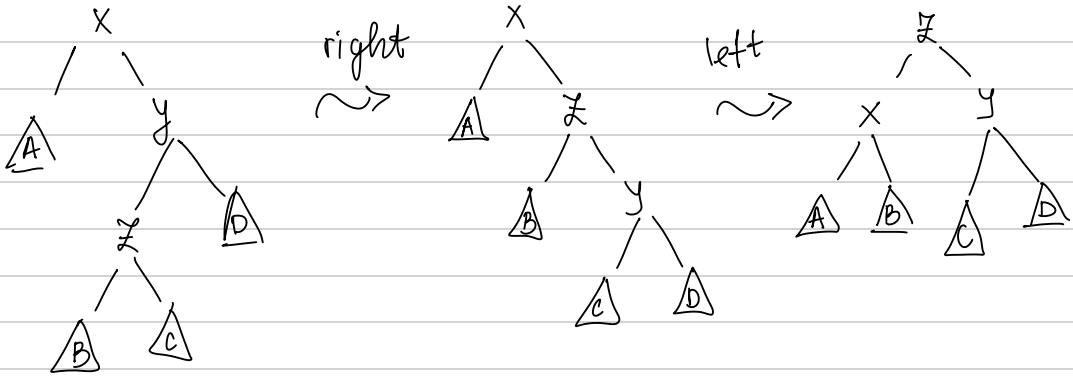
Rotations in general:



righ






X
A
Y
Z
D
B
C
right
X
A
Z
B
Y
C
D
left
Z
X
Y
A
B
C
D