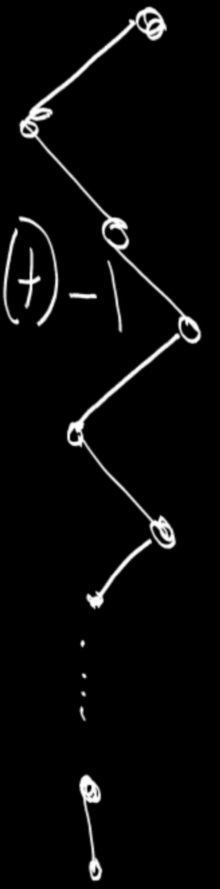
Note: for contains (as for adding/removing elements),
the complexity is

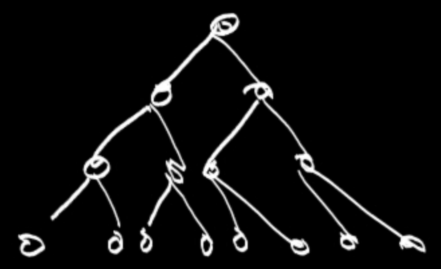
- Θ(size) in a BT
- Θ(height) in a BST

Q: relation height/size?

worst case: tree is degenerated: height(t) = size(t) − 1



- best case (perfect tree)



h	size
1	3
2	7
3	15
...	
h	$2^{h+1} - 1$

$size(t) = \Theta(2^{height(t)})$

$\Rightarrow height(t) = \log(size(t))$

Average case! $height(t)$ is still $\Theta(\log(size(t)))$

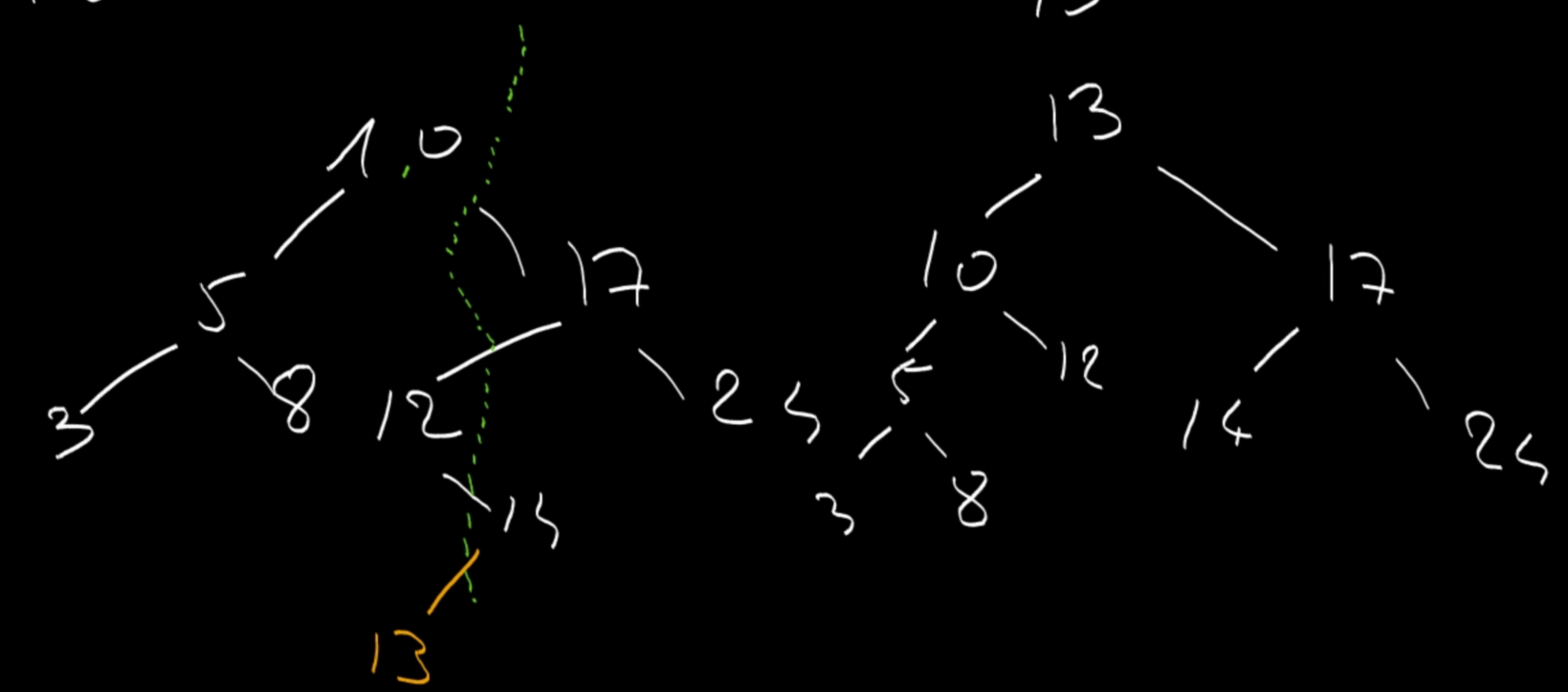
1 → 3 → 5 → 7 → ...

Important quantity/measure: average depth of nodes in the tree

usually, $\overline{f}(n) = \sum_{n \in E} f(n) \Big/ card(E)$

$$\overline{h}(t) = \frac{PL(t)}{size(t)}$$

Root insertion in a BST



13?

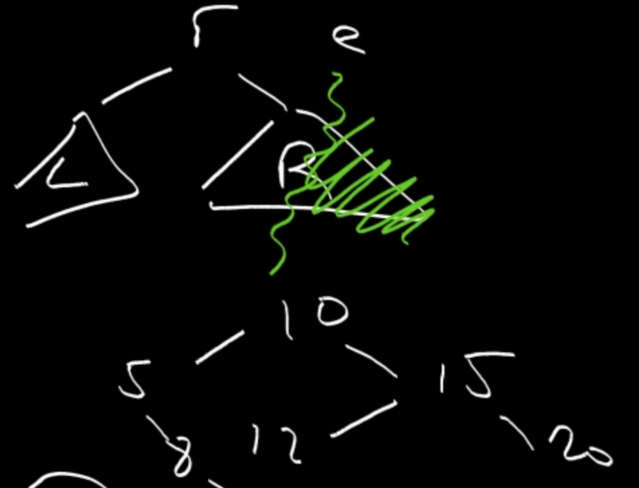
ADT extension BST

Operations

lessThan : BST × Element → BST

greaterThan : BST × Element → BST

rootAdd : BST × Element → BST



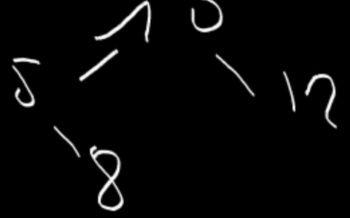
Axioms

A. lessThan(new, e) ≡ new

A_2 lessThan(⟨r, L, R⟩, r) ≡ L

A_3 (r ≤ e) lessThan(⟨r, L, R⟩, e) ≡ lessThan(L, e)

A_4 lessThan(⟨r, L, R⟩, e) ≡ ⟨r, L, lessThan(R, e)⟩



lessThan(10, 13)

A_4 ⟨10, ⟨5⟩, lessThan(15, 13)⟩

A_3 ⟨10, ⟨5⟩, lessThan(12, 13)⟩

A_4 ⟨10, ⟨5⟩, ⟨12, new, lessThan(new, 13)⟩⟩

A_1 ⟨10, ⟨5⟩, ⟨12⟩⟩

A_5 greaterthan(new, e) ≡ new

A_6 greaterthan(<r, L, R>, r) ≡ R

A_7 e > r ⇒ greaterthan(<r, L, R>, e) ≡ greaterthan(R, e)

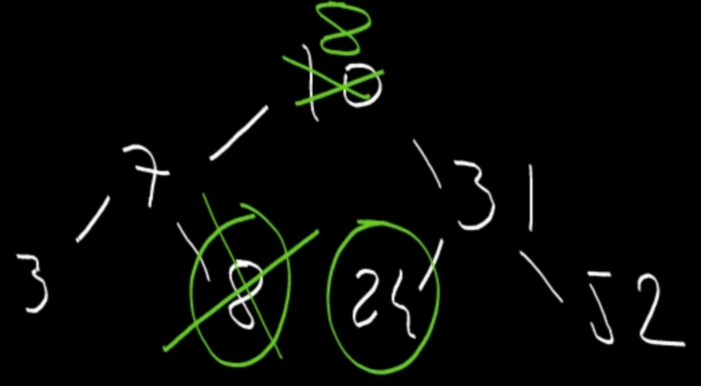
A_8 greaterthan(<r, L, R>, e) ≡ <r, greaterthan(L, e), R>

A_9 rootAdd(t, e) ≡ < e, lessthan(t, e), greaterthan(t, e)>

Removal in a BST

Different cases:

1) the element to remove is a leaf : just remove it
2) _____ has only one child : replace with the child
3) if 2 children : 2 possibilities → smallest of right child
   → greatest of left child



2 companion operations

- max(t) → greatest elt in t
- delMax(t) → removes the greatest elt in t

ADT extension BST

Operations

$max : BST \longrightarrow Element$

$delMax : BST \longrightarrow BST$

$remove : BST \times Element \longrightarrow BST$

Pre condition

$max(t)$ if $t \not\equiv new$

Axioms

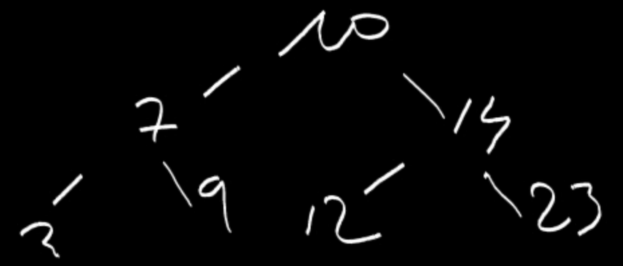
A_1 $max(\langle r, L, new \rangle) \equiv r$

A_2 $max(\langle r, L, R \rangle) \equiv max(R)$

A_3 $delMax(new) \equiv new$

A_4 $delMax(\langle r, L, new \rangle) \equiv L$

A_5 $delMax(\langle r, L, R \rangle) \equiv \langle r, L, delMax(R) \rangle$

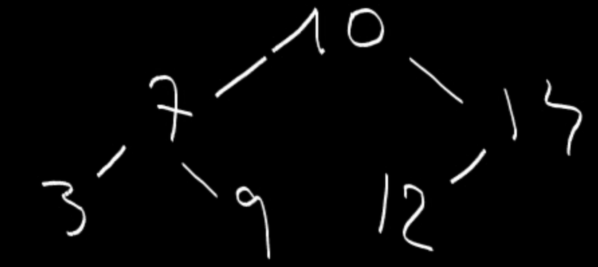


$delMax(10)$

$A_5 < 10, 7, delMax(14) >$

$A_5 < 10, 7, < 14, 12, delMax(23) >>$

$A_4 < 10, 7, < 14, < 12 >, new >>$



A_6 $remove(new, e) \equiv new$

A_7 $e < r \Rightarrow remove(<r, L, R>, e) \equiv <r, remove(L, e), R>$

A_8 $e > r \Rightarrow remove(<r, L, R>, e) \equiv <r, L, remove(R, e)>$

~~A_9 $remove(<r, new, new>, r) \equiv new$~~

A_9 $remove(<r, L, new>, r) \equiv L$

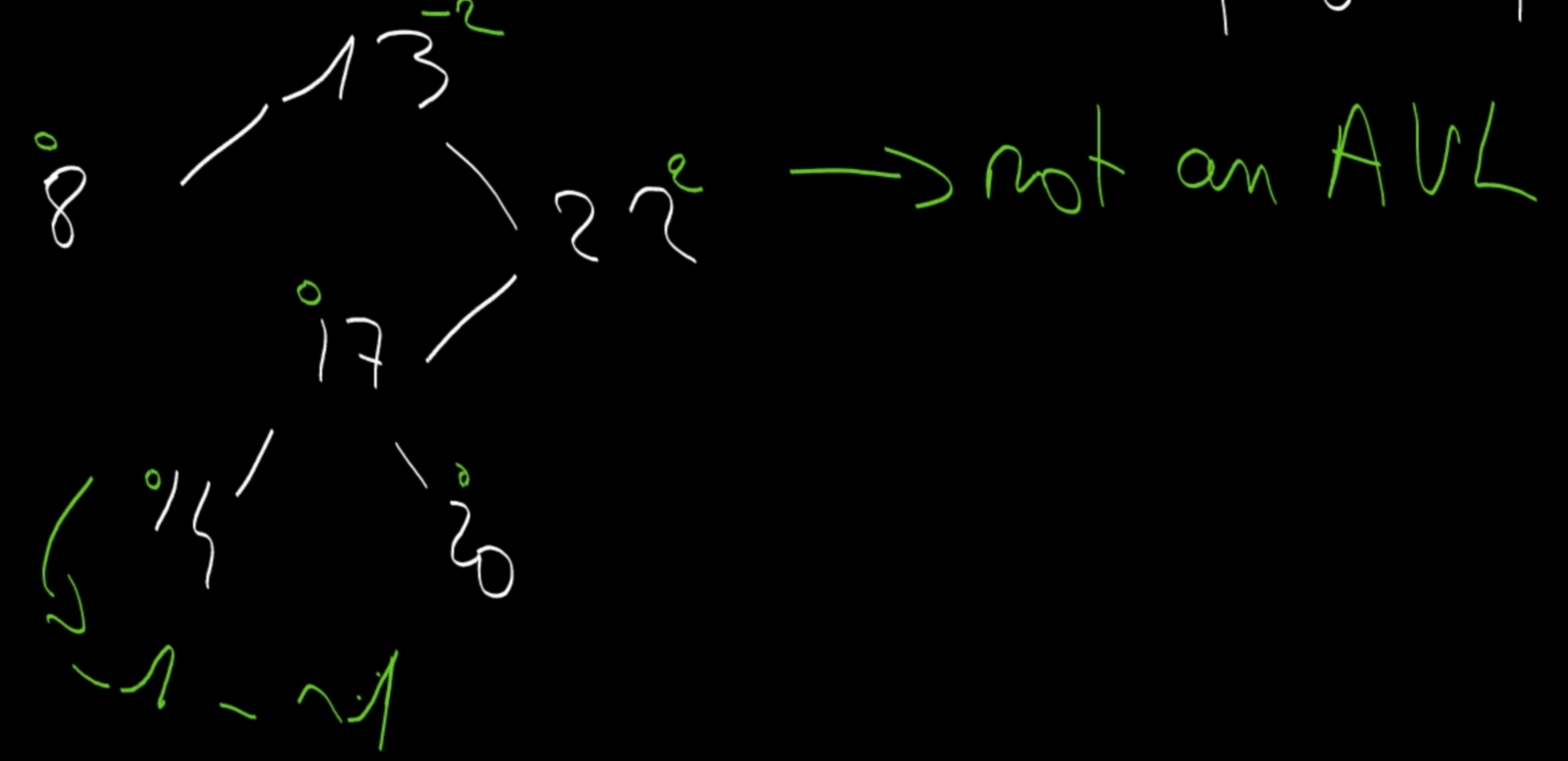
A_{10} $remove(<r, new, R>, r) \equiv R$

A_{11} $remove(<r, L, R>, r) \equiv <max(L), delMax(L), R>$

AVL (Adelson-Velskii & Landis)

def: degree of a node in t $=_{def}$ height(left(node)) − height(right(node))

AVL : BST $\wedge$ $\forall n \in t$ $|deg(n)| \leq 1$



→ not an AVL

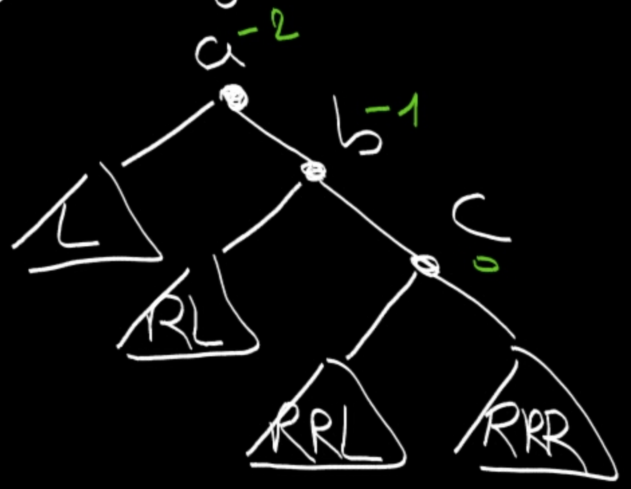
How to keep the tree "balanced" (degrees in $\{-1, 0, 1\}$)?

Context: an addition / removal brought a degree of ± 2 for one of the nodes $\longrightarrow$ rotation(s) of the tree

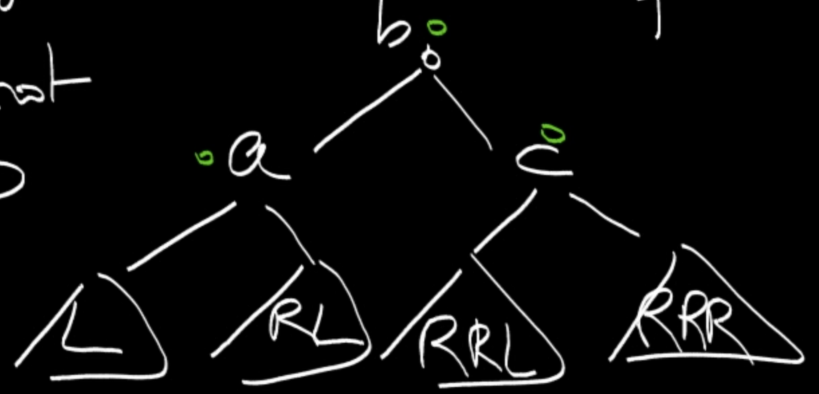
4 kinds of rotations

1) $deg(t) = +2$ and $deg(left(t)) = +1 \rightarrow$ right rotation

2) deg(t) = -2 _ deg(right(t)) = -1 ⟶ left rotation

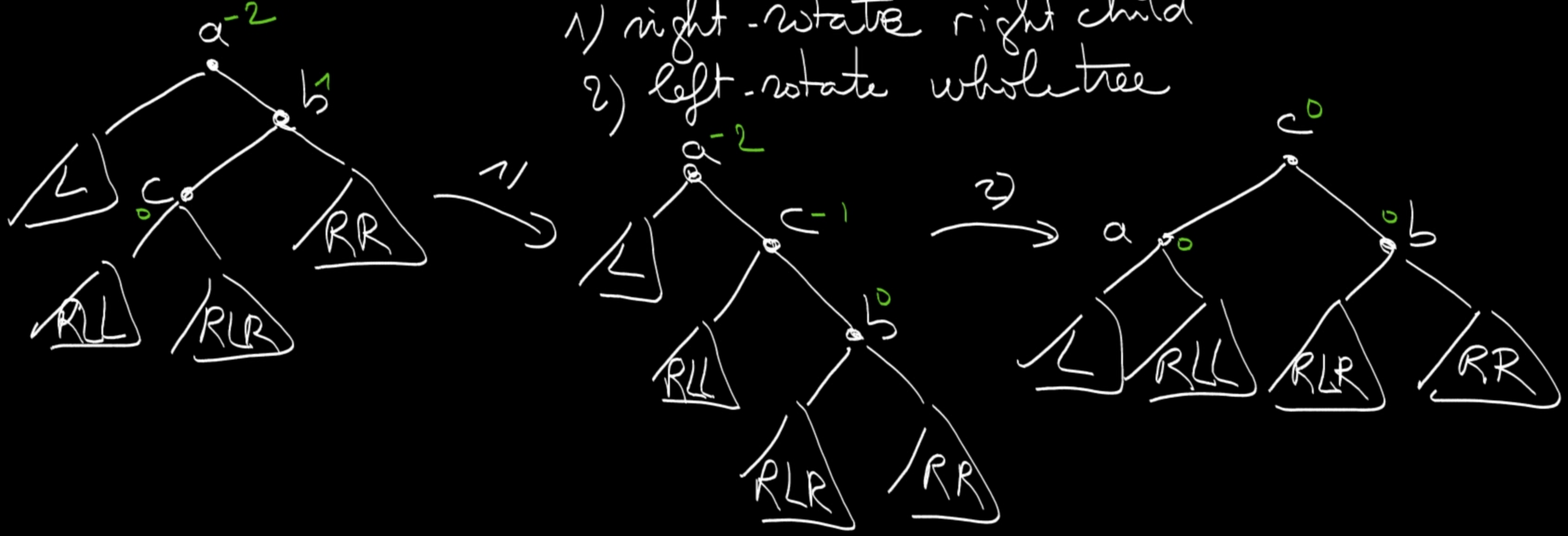


left rot ⟶



3) $deg(t) = -2$, $deg(right(t)) = +1$ → right left rotation

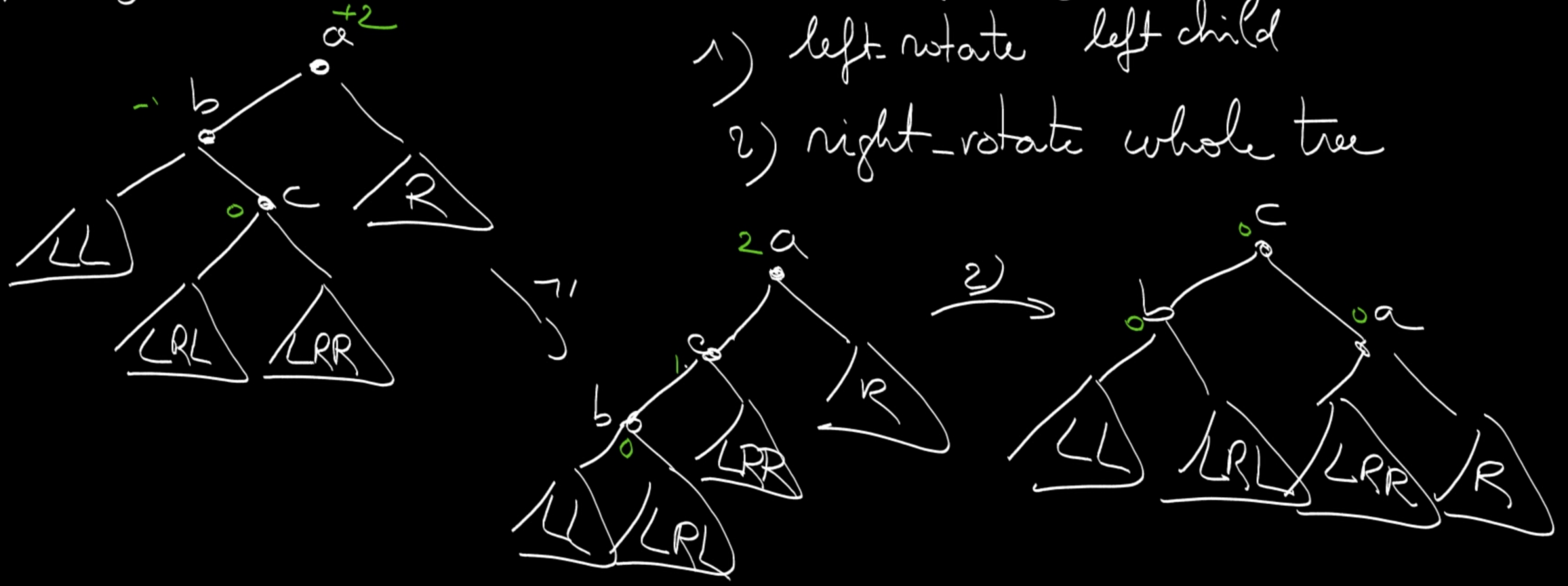
1) right-rotate right child
2) left-rotate whole tree



4) $deg(t) = +2$, $deg(left(t)) = -1$ → left_right rotation

1) left_rotate left child

2) right_rotate whole tree



exercise: starting from an empty BST, add values

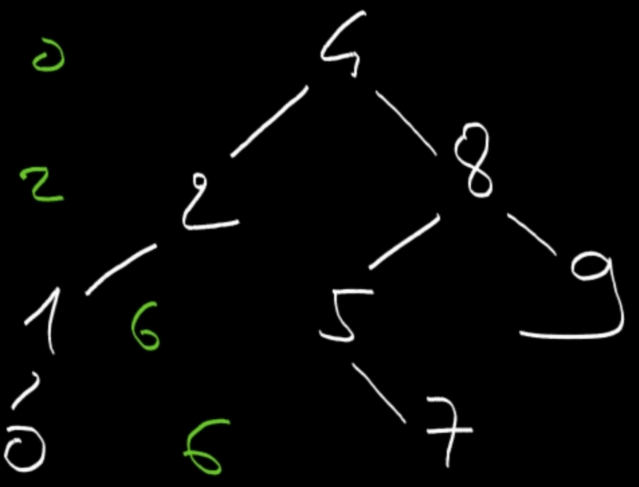
4 2 8 5 7 1 0 9

1) using leaf addition

2) using root addition

3) using AVL addition (leaf add + balance)

Leaf addition :



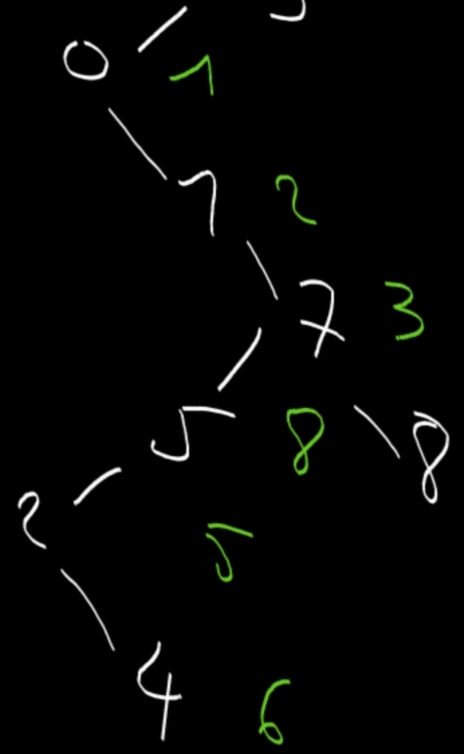
$h = 3$

$PL = 14$

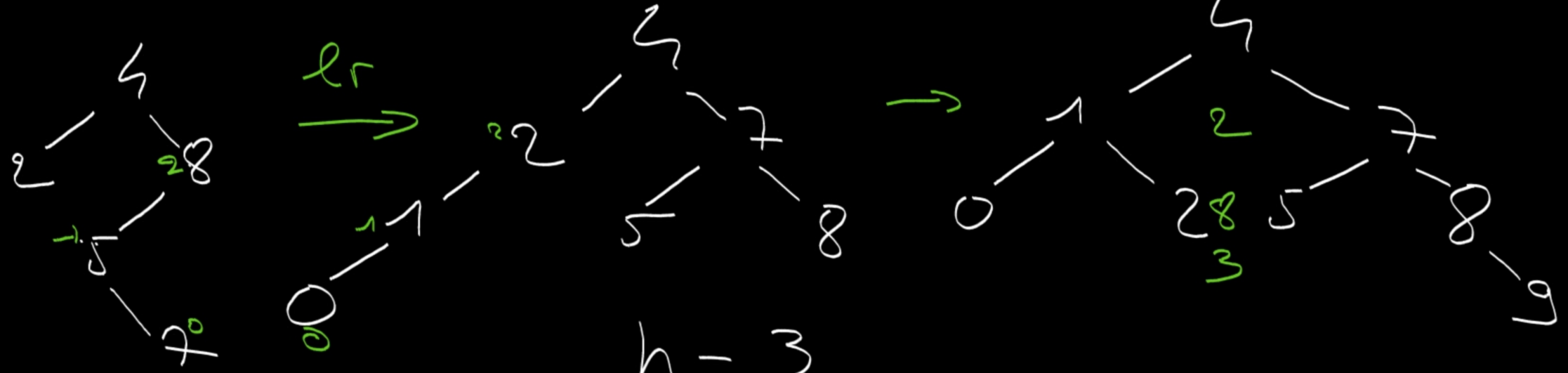
Root addition 8

$h = 6$

$PL = 25$



"AVL" addition



$h = 3$

$PL = 13$

ADT AVL extends BST

Operations

lrot : BST → AVL

rrot : BST → AVL

lrrot : BST → AVL

rlrot : BST → AVL

Preconditions:

lrot(t) iff t ≠ new and right(t) ≠ new and right(right(t)) ≠ new

rrot(t) iff t ≠ new and left(t) ≠ new and left(left(t)) ≠ new

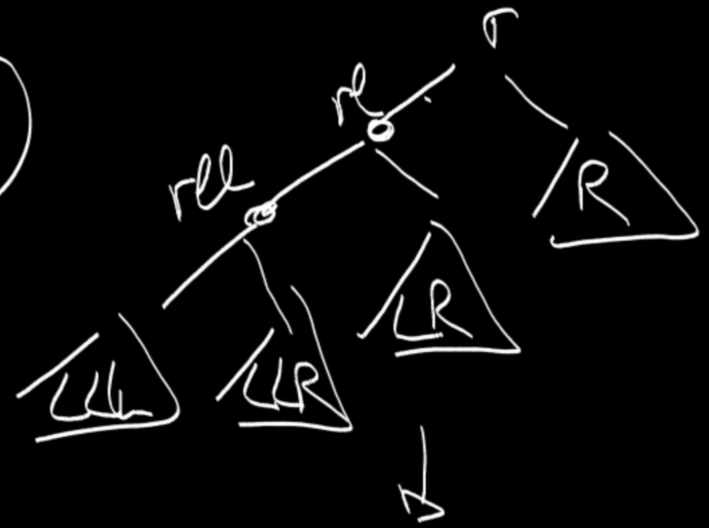
lrrot(t) iff t ——— right(left(t)) ≠ new

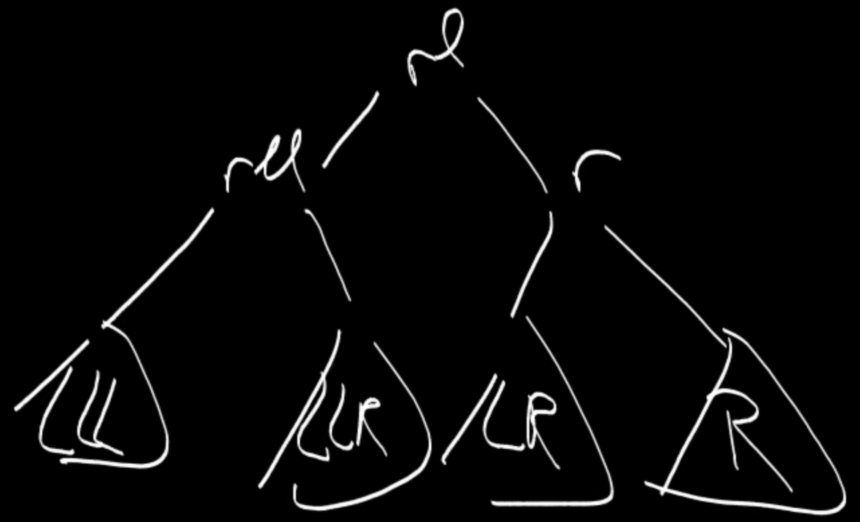
rlrot(t) ——— right(t) ——— left(right(t)) ≠ new

A. $rot(<r, <rl, <rll, LLL, LLR>, LR>, R>)$

$\equiv <rl, <rll, LLL, LLR>, <r, LR, R>>$

(2) (1)





A_2 $\text{lrot}(\langle r, \langle lr, LL, \langle lrr, LRL, LRR\rangle\rangle, R\rangle)$

$\equiv \langle lrr, \langle lr, LL, LRL\rangle, \langle r, LRR, R\rangle\rangle$

https://tiny.one/algo56