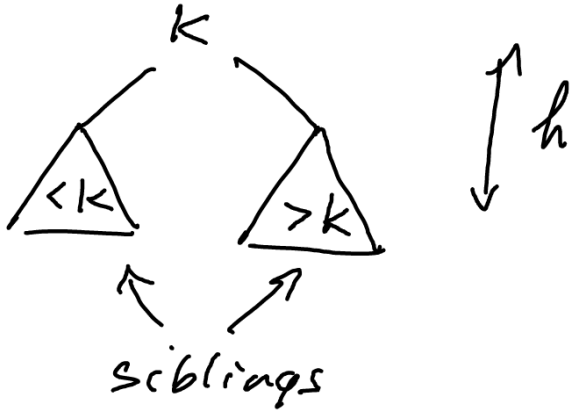
Balanced Binary Search Trees : AVL Trees

Binary Search Tree invariant:
- all nodes in left subtree < k
- " right " > k
- left & right subtrees are BSTs.



⇒ add, search, delete are O(h)

How to ensure that h is O(lg n) — tree is balanced?

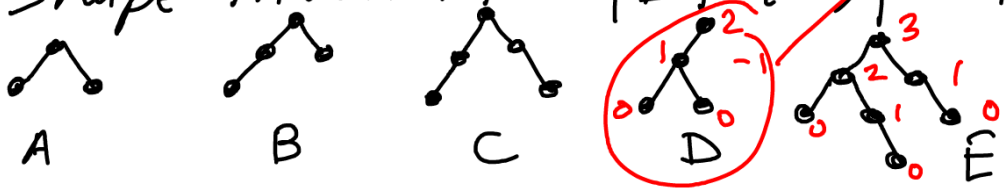
AVL tree — the original balanced tree
— idea: sibling subtrees are close in height

Balance factor of node t

$BF(t) = height(t.left) - height(t.right)$

AVL Shape invariant: $|BF(t)| \leq 1$ for all nodes t

AVL trees?

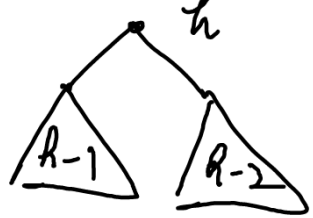


$|BF(t)| \leq 1$ guarantees h is $O(\lg n)$?

h too large for n $\iff$ n is too small for h.

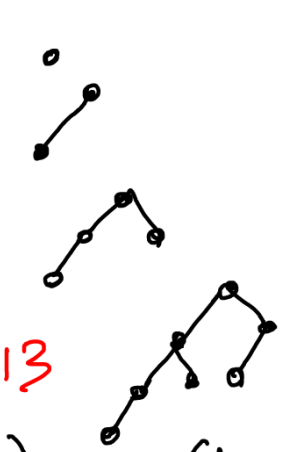
Most unbalanced tree has minimum number of nodes for its height h.

$BF(t) = 1$ at all t.



let $N(h)$ be minimum # of nodes at height h.

$N(0) = 1$
$N(1) = 2$
$N(2) = 4$
$N(3) = 7$
$N(4) = 12$



$N(h) = N(h-1) + N(h-2) + 1$

A: 1
B: 2
C: 3
D: 4
E: 5

$N(h) = F_{h+3} - 1$

(F_i'bonacci numbers!)

$F(h) = \frac{1}{\sqrt{5}}\left(\phi^h - \left(\frac{-1}{\phi}\right)^h\right) \to 0$

ϕ = Golden Ratio ≈ 1.618

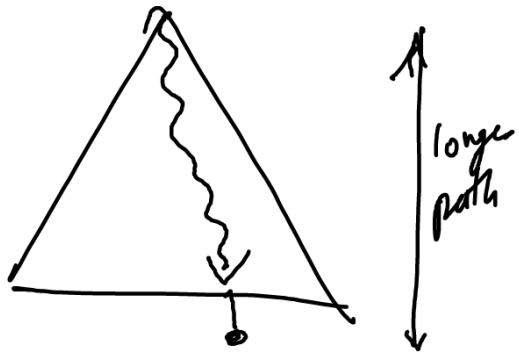
$-1/\phi \cong -.618$

$F_1 = 1$
$F_2 = 1$
$F_3 = 2$
$F_4 = 3$
$F_5 = 5$
$F_6 = 8$
$F_7 = 13$
...

$$n \geq c\phi^{h} - 1$$

$$h \leq \log_{\phi}\left(\frac{n+1}{c}\right) = O(\log n)$$

AVL insertion (add)

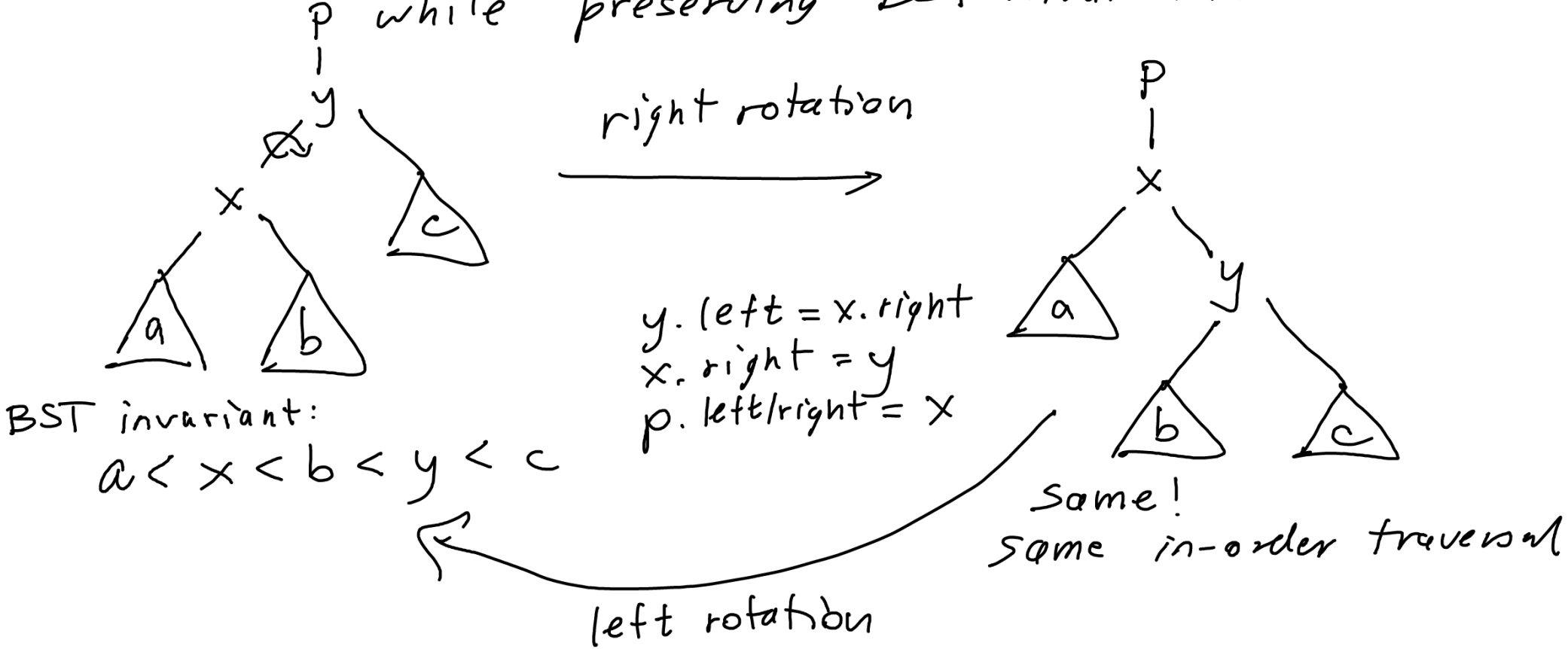


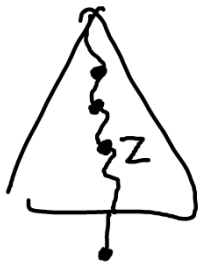
1. Do usual BST insertion at a leaf

May unbalance nodes on the path to new leaf BF=2 or −2

idea: use tree rotations to fix shape invariant.

Tree rotation : reorg. nodes while preserving BST invariant.

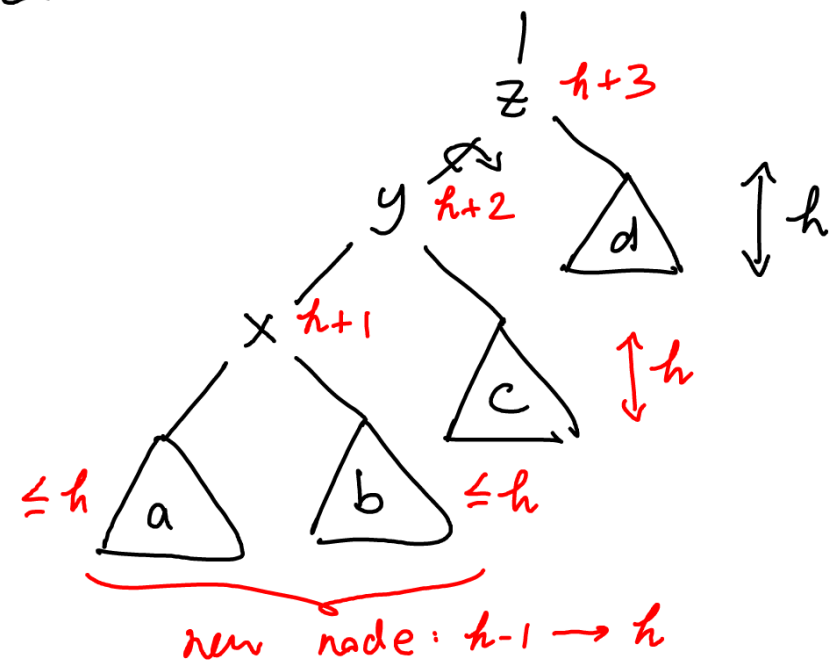




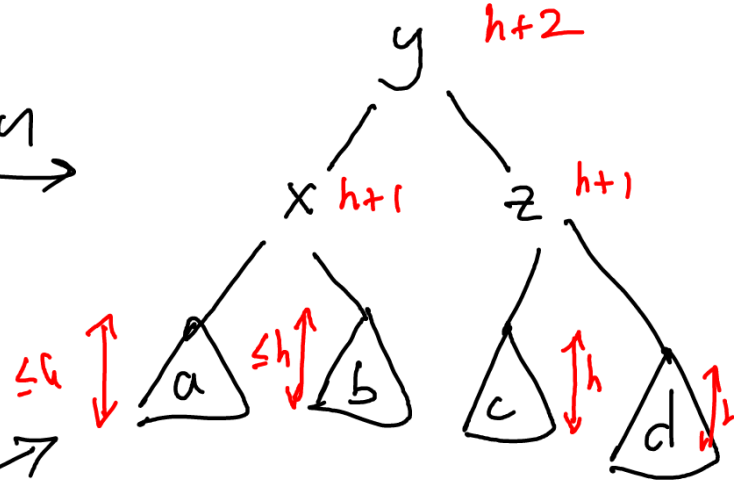
let z be lowest unbalanced node.

Four cases: BF = 2 ×2
BF = -2 ×2

1. case LL: BF = 2

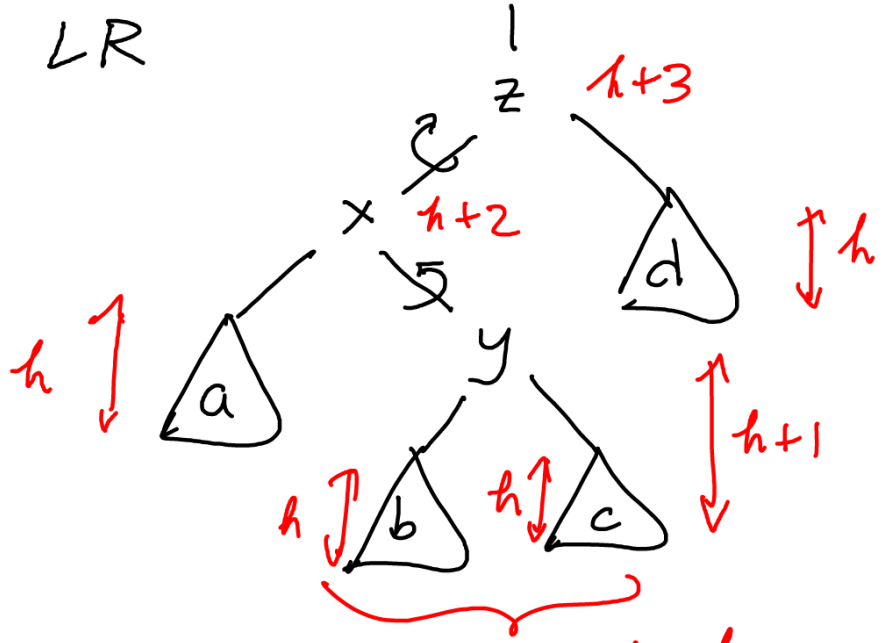


right rotation

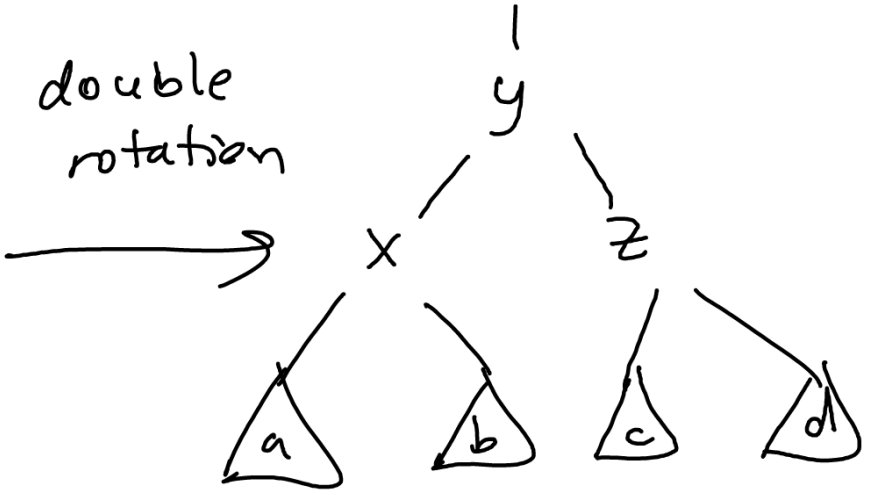


BST invariant: $a < x < b < y < c < z < d$

2. case LR



double rotation



update x & y & z & parent

3, 4: case RR, RL: symmetric. (right subtree is too tall)

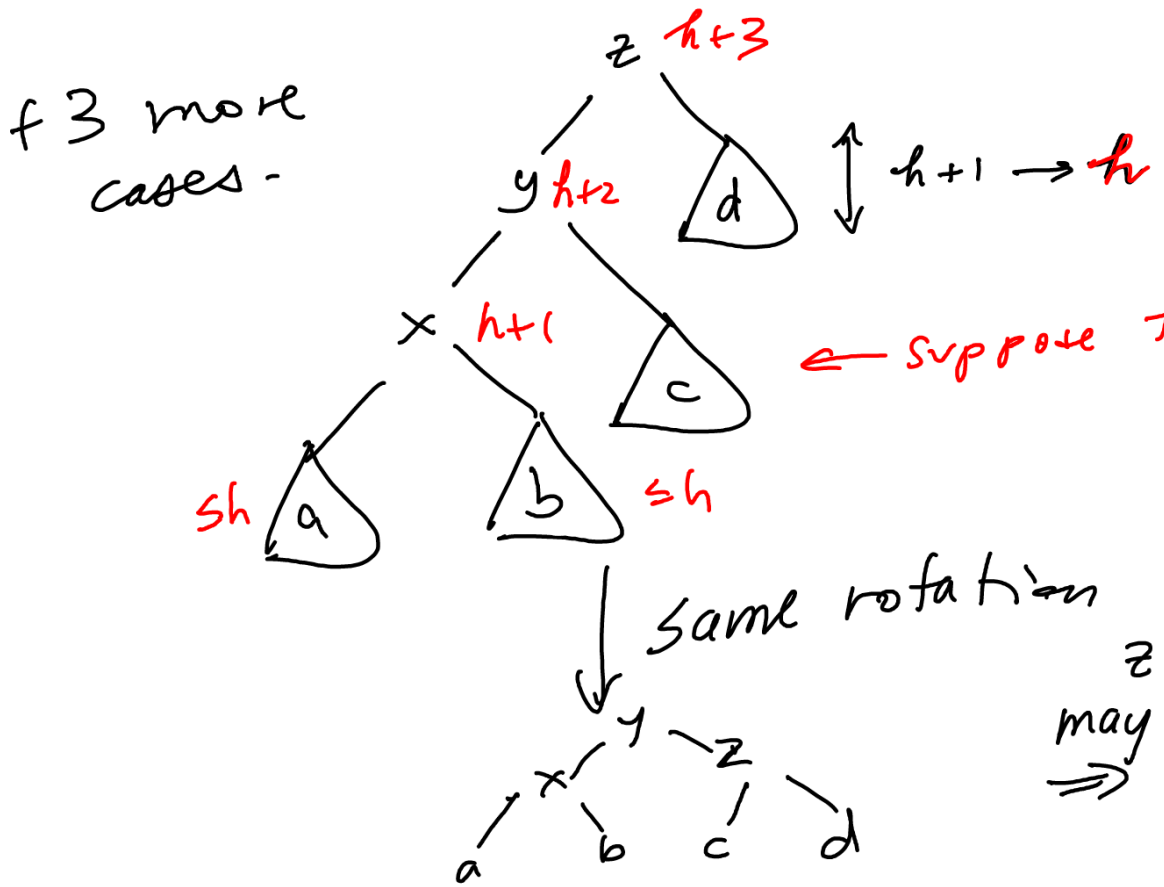
BF = -2

Details:

- need to keep track of heights & update them during balancing
- can be recursive or iteratively
- can keep track of BF instead of height

AVL deletion

let z be lowest unbalanced node.
If BF=2



if 3 more cases.

← suppose that only x is at h+1
height(c) = h.

same rotation as LL case.
z height: h+3 → h+2
may unbalance ancestors of z
⇒ walk up and rebalance ancestors as necessary.
(recursive convenient)

Other balanced trees

Red-Black trees : red / black.

2-3, 2-3-4, B trees: used in databases & filesystems

N-ary: smaller h.

Splay trees : network routers

- no invariant
- amortized $O(\lg n)$ time & good locality.