Runway Reservation Problem

- Single runway in an airport.
- Authority maintains landing times of each flight.
- When a new request comes with time t, checks if there is any landing reserved between $(t-3)$ to $(t+3)$. If not approve. O/W don't approve.

new_request(t)

if (- - - - -)

- Condition Checking
- Insert

O/W

Approve

Don't Approve

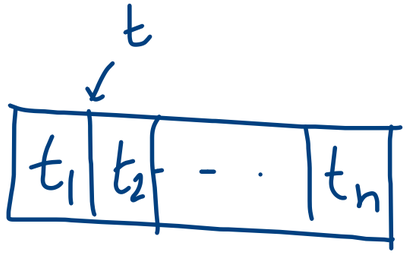
$T = \{t_1, \ldots, t_n\}$

① Array:

(unsorted)

$O(n)$

(sorted)



Binary Search

↳ $O(\log n)$

Insertion

→ $O(n)$

② Linked List

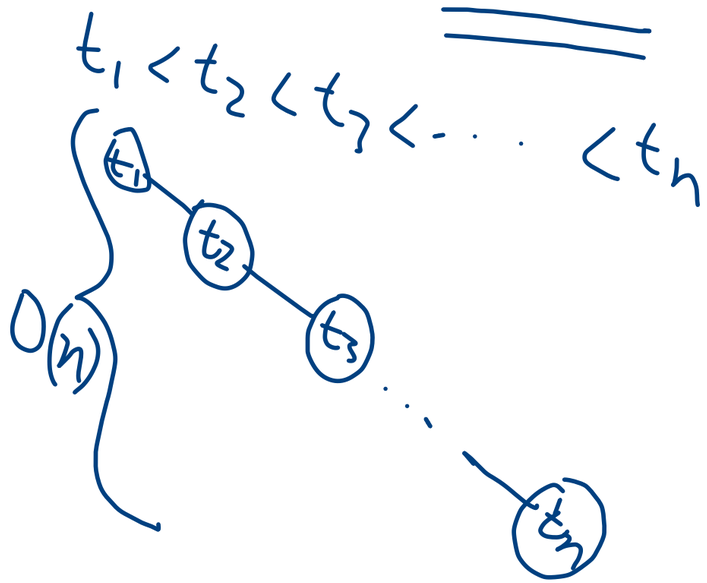
$O(n)$

③ Search Binary Tree

(t1)

Searching in BST

↳ O(h) or O(log n)?



Condition → Searching in BST
⇒ O(h)

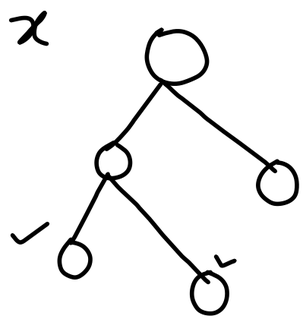
Insert → Insertion in BST
⇒ O(h)

Certain instances
h ≈ O(n)

Balanced Binary Search Tree

$h = O(\log n)$

AVL Tree (Adelson, Velsky, Landis)



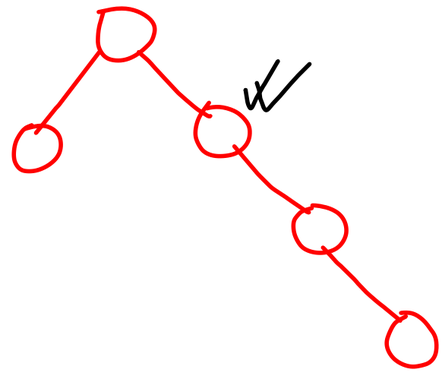
height(x) = 2

height of node → max distance from that node to any leaf.

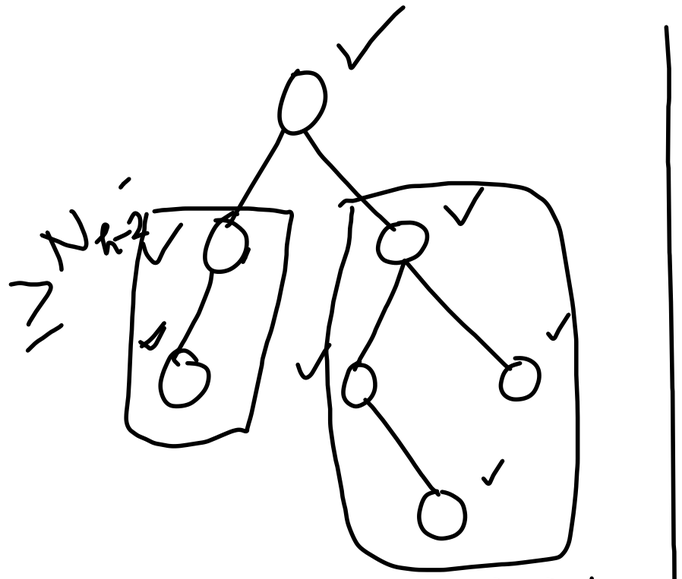
$\forall$ node x,

AVL: $\forall x$, $\left| \text{height}(lc(x)) - \text{height}(rc(x)) \right| \leq 1$

$$\text{height}(x) = \max\{\text{height}(lc(x)), \text{height}(rc(x))\} + 1$$



Not AVL



AVL

$$N_h = 2^{\frac{h}{2}} N_0 + 2^{h/2}$$

$$n = 2^{\frac{h}{2}+1}$$

$$h = \log(n)$$

$$h = O(\log n)$$

Height of an AVL tree is $O(\log n)$

N_h = # nodes possible with an AVL tree with height h.

$$N_h \geq 2N_{h-2} + 1$$

$$\geq 2(2N_{h-4}+1) + 1$$

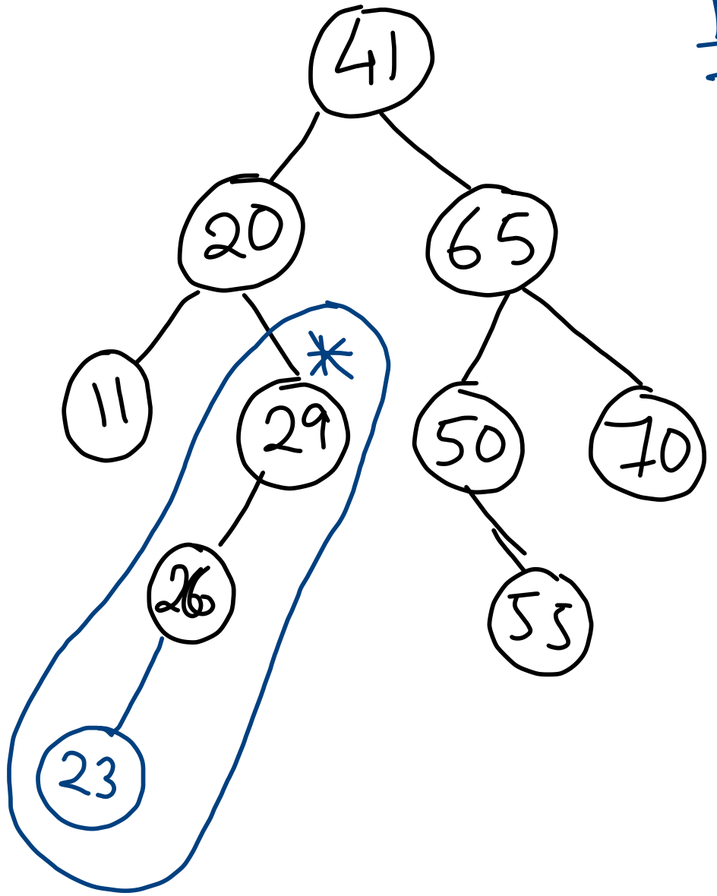
$$= 2^2 N_{h-4} + (1+2)$$

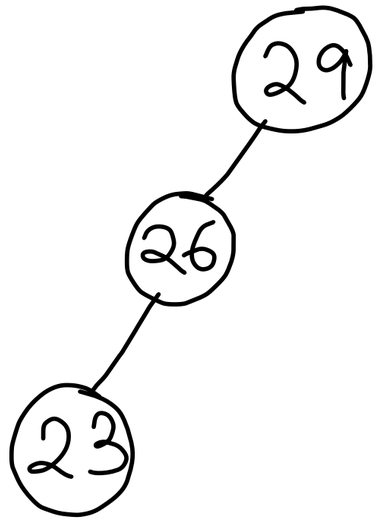
$$\vdots$$

$$= 2^i N_{h-2i} + (1+2+\cdots+2^{i-1})$$

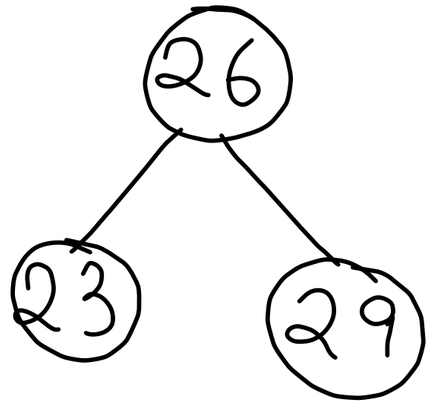
$$(1+2+\cdots+2^{i-1}) = (2^i - 1)$$

Insert(23)

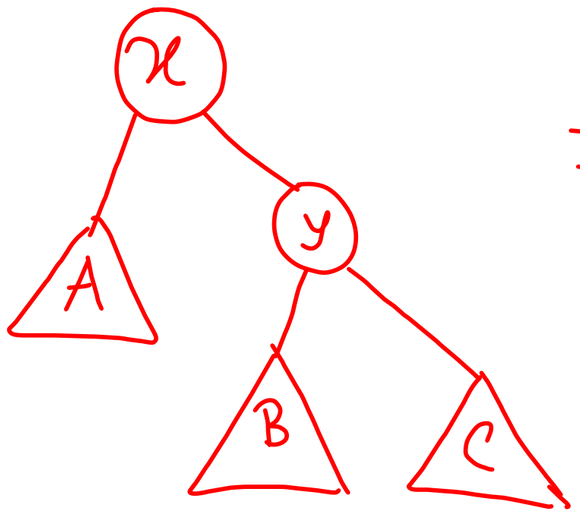




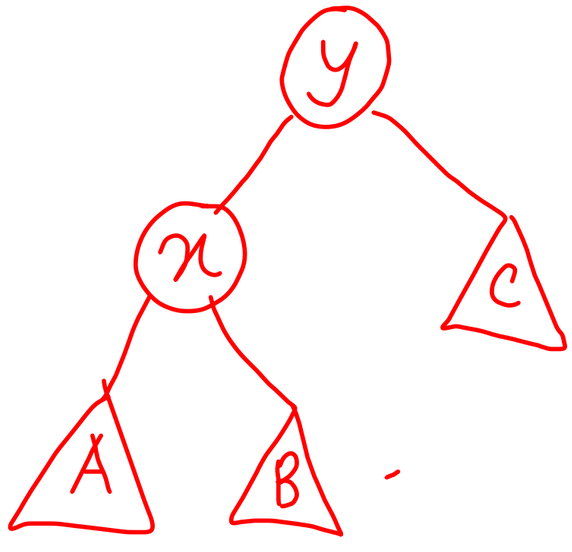
(Rotation)



Rotation



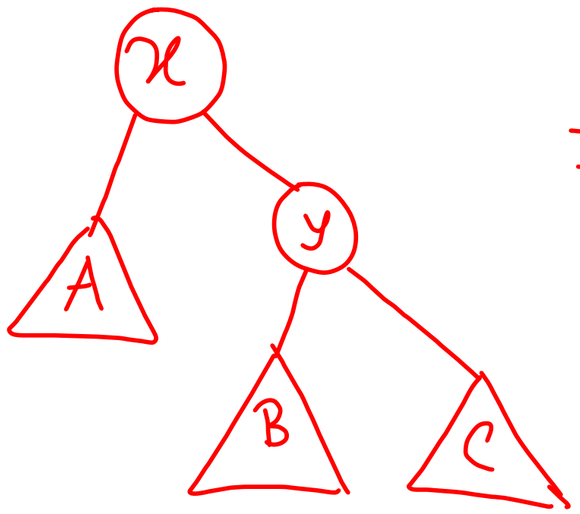
Left Rotation on x



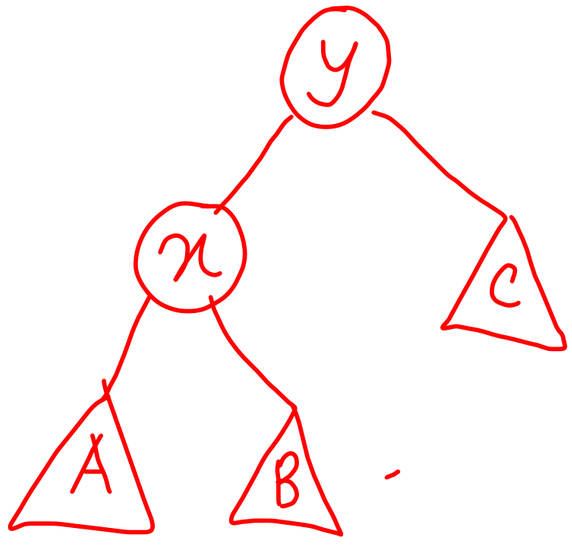
$A < x < B < y < C$

$A < x < B < y < C$

Rotation



Left Rotation on x



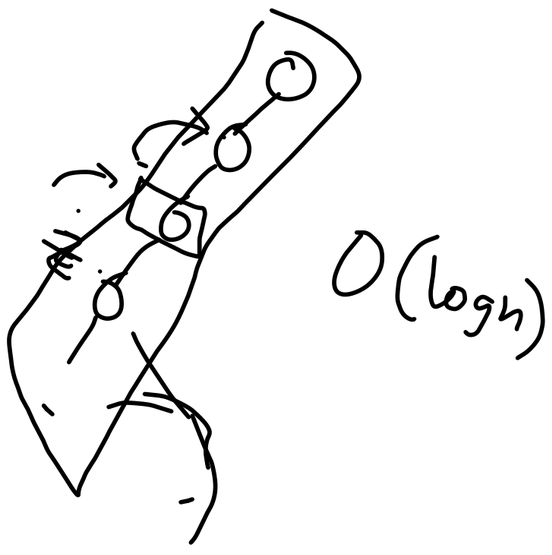
$A < x < B < y < C$

$A < x < B < y < C$

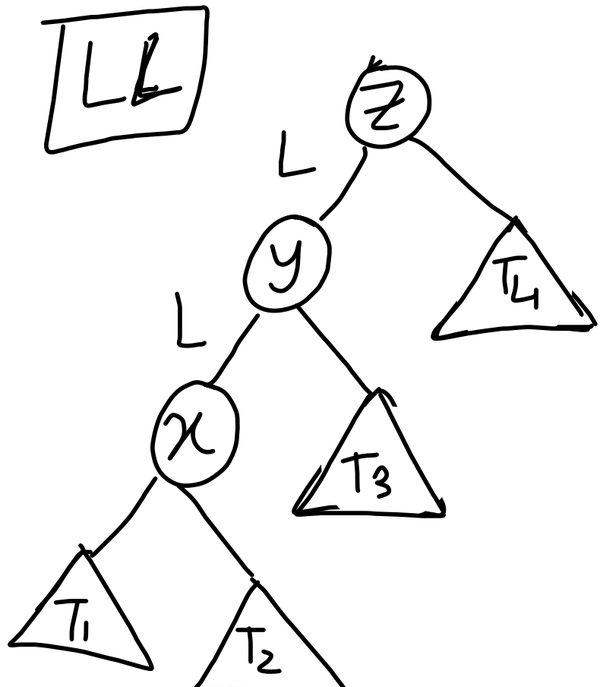
AVL Insertion

$O(\log n)$

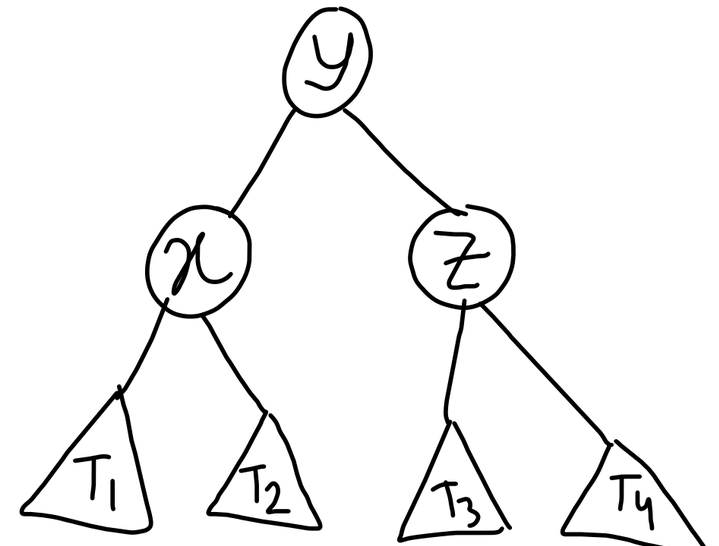
- Insert (w)
- First element in the path from w to root which is unbalanced
  Let's call that element (z)
- Claim: z has a child & a grand-child in the path from z to w (may include w itself)
  ⇓ y ⇓ x



LL

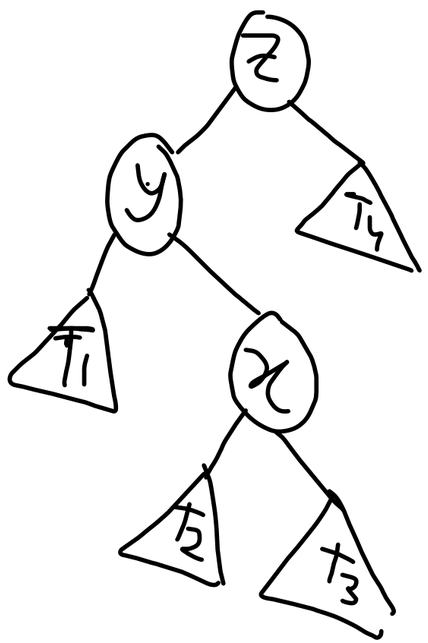


RR(z)

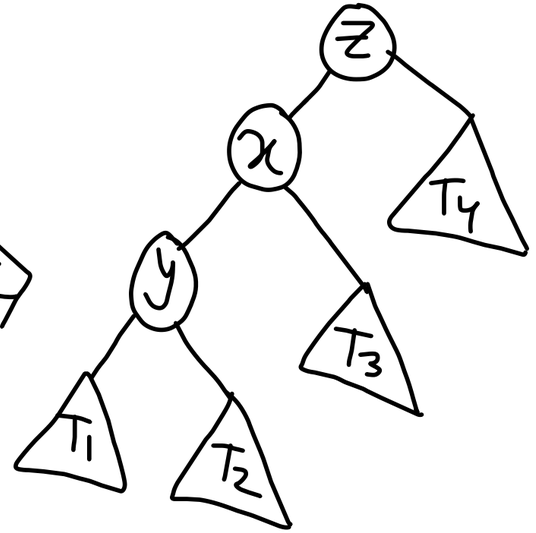


O(log n)

LR



LR(y)



RR(z)