# Lect-10: Binary Balanced Trees

Goal: Support add/remove to maintain small height

$\rightarrow h = O(\log n)$ — $n$: #nodes/elements in Binary Search Tree

## §1. Definition of Height Balance

$v \rightarrow$ a node in Binary tree

$h(v) \rightarrow$ maximum distance to descendent leaf (height)

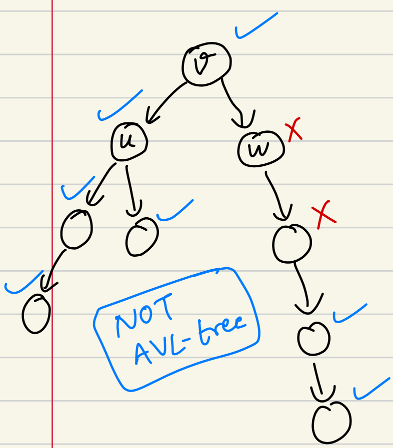
Convention: $h(\text{null}) = -1$.

**Observation:** if $u = v.\text{left}$, $w = v.\text{right}$

, then $$\boxed{h(v) = 1 + \max(h(u), h(w))}$$

**Def:** $v$ is (height) balanced if height of $v$'s children differ by at most 1.

$$\boxed{|h(u) - h(w)| \leq 1}$$




$\Rightarrow$ We can say a binary tree $T$ is (height-balanced) or AVT-tree if all nodes are height balanced.

## §2 Benefits of Balance

**Goal:** If $T$ is balanced, (AVL) then its height $h$ is $O(\log n)$
($n = \#$ of nodes)

**Roundabout Method:** Consider $m(h) =$ min # of nodes in an AVL-tree of height $h$.

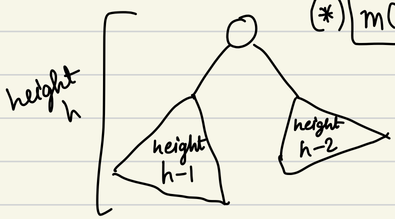
- If $n \leq m(h)$, then height is atmost $h$.

→ what is $m(h)$ for small values? $\left[\begin{array}{l} m(0) = 1 \\ m(1) = 2 \\ m(2) = 4 \end{array}\right]$

→ what can we say about structure of minimal (# of nodes) AVL-tree of height $h$? $(h \geqslant 2)$

(*) $\boxed{m(h) = 1 + m(h-1) + m(h-2)}$

$\Rightarrow (m(1) < m(2) < \cdots\cdots)$




$$m(h) = 1 + m(h-1) + m(h-2)$$
$$m(h) > 2 \cdot \underbrace{m(h-2)}_{> 2 \cdot m(h-4)}$$

$$m(h) > 4 \cdot m(h-4) > 2^i \cdot m(h-2i)$$

Setting $i = \left\lceil \frac{h}{2} \right\rceil - 1$ ← "round up" $\Rightarrow h - 2i = 0$ or $1$.

So, $m(h-2i) = 1$ or $2 \geqslant 1$

$$m(h) > 2^{\lceil \frac{h}{2} \rceil - 1} \cdot \underbrace{m(0 \text{ or } 1)}_{\geqslant 1}$$

$$m(h) \geqslant 2^{\lceil \frac{h}{2} \rceil - 1} \Rightarrow \boxed{m(h) \geqslant 2^{\frac{h}{2} - 1}}$$

$$\log(m(h)) \geq \left(\frac{h}{2} - 1\right)$$

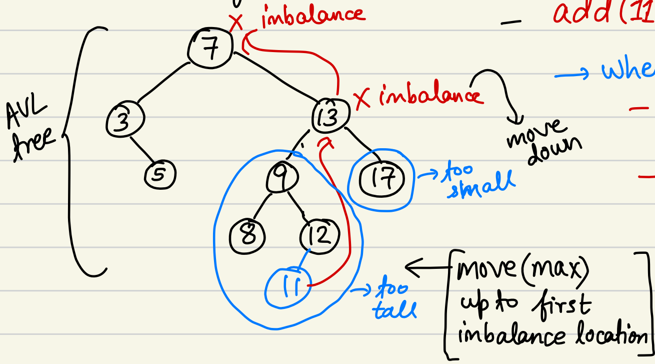
$$h \leq 2(1 + \log(m(h)))$$

**Conclusion:** If $T$ is AVL tree with $n$-nodes and height $h$, then

$$h \leq 2\log(m(h)) + 2 \leq 2\log(n) + 2$$

So, $\boxed{h = O(\log n)}$ //

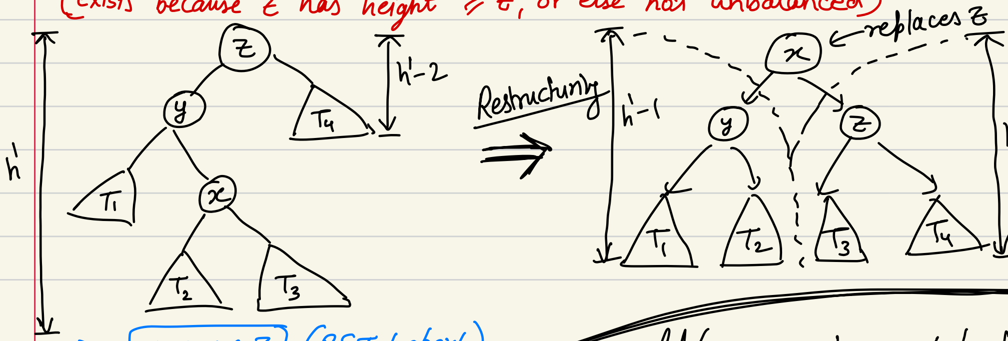
## §3 Maintaining Balance




- add(11) – no longer balanced
  → where can the imbalance occur?
  - only at ancestors of added node
  - Only $\leq h = O(\log n)$ many of there.

$w$ = added node, which is causing imbalance [eg. 11]
$z$ = deepest node where imbalance occurs. [eg. 13]
$y$ = child of $z$ in direction of $w$ [eg. 9]
$x$ = child of $y$ in direction of $w$ [eg. 12]

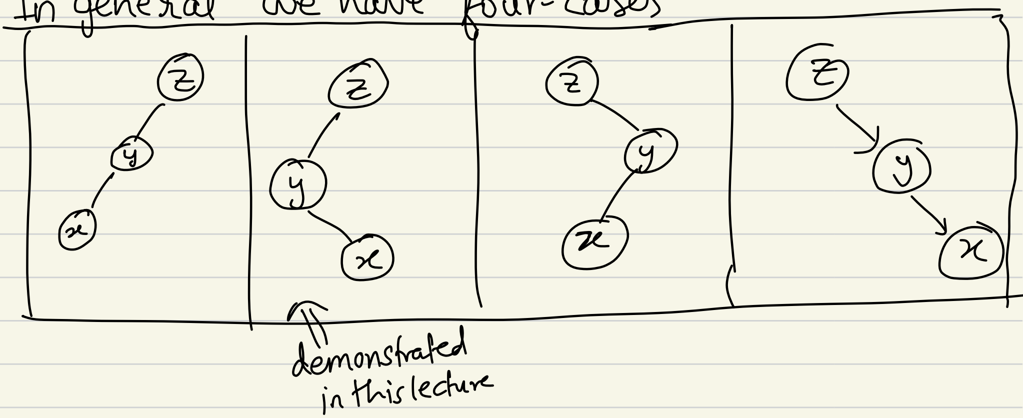
(exists because $z$ has height $\geq 2$, or else not unbalanced)




$\Rightarrow \boxed{y < x < z}$ (BST property)

add/remove in next lecture.

In general we have four-cases




Q1 Why does it maintain the BST property?

Q2 Why it restores balance?

Q3 Running Time?