# 2024/09/10 - Data Structures - Week 07

## AVL Trees

For binary search tree :

time complexity of search, insertion, deletion $\Big\}$ $O(h)$

$h = O(n)$ $\quad$ $n :=$ no. of nodes in BST

- as bad as linked list if $h = O(n)$

we need $h$ to be $O(\log_2 n)$

Modify BST $\longrightarrow$ get $h$ in $O(\log_2 n)$

# AVL Tree

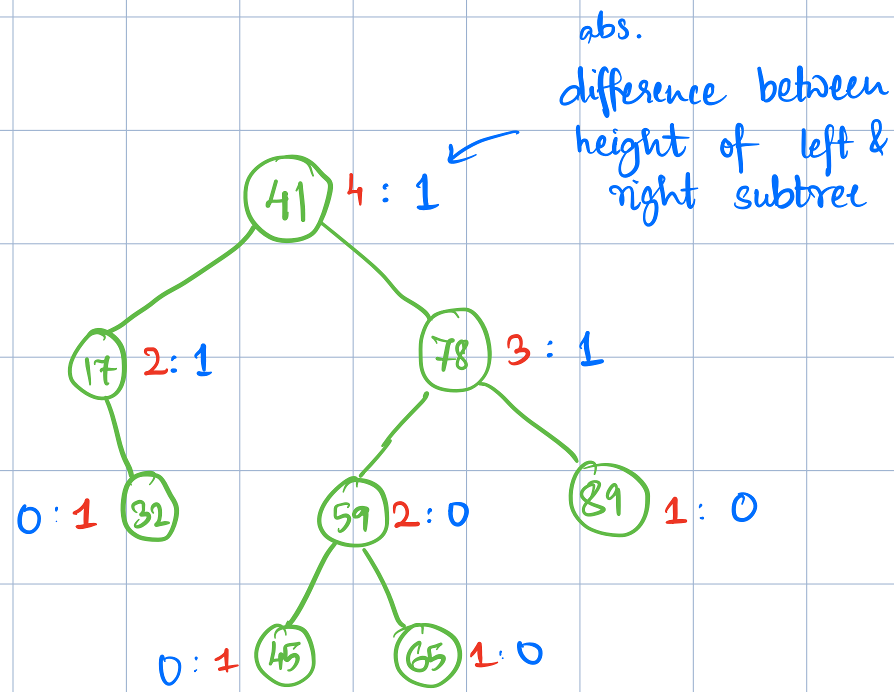
(i) BST

(ii) For every internal node in BST, difference b/w height of left subtree and right subtree is atmost 1.

leaf node is at height 1.

height ⟷ global parameter

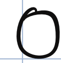
condition (ii) ⟷ local



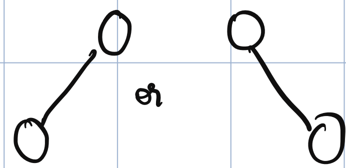

AVL tree example.

**Theorem:** The height of AVL tree of $n$ nodes is $O(\log n)$.

**Proof:** $n(h) :=$ minimum number of nodes in an AVL tree of height $h$.

$n(1) = 1$  leaf node

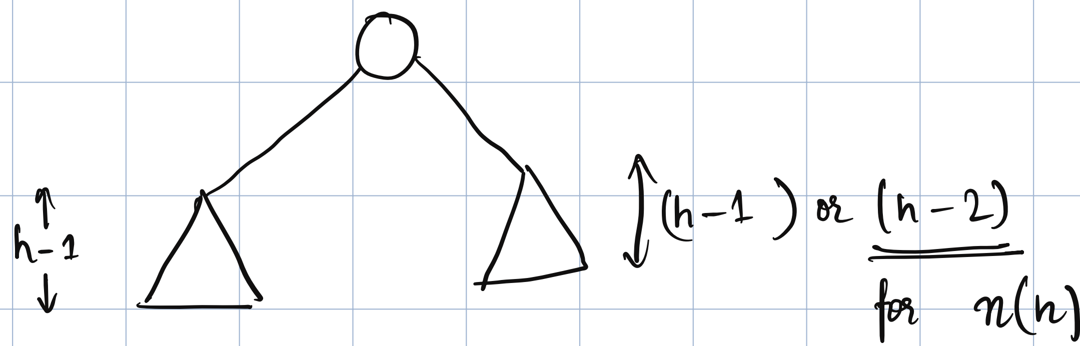
$n(2) = 2$




For $h \geq 3$: an AVL tree consists of

if more than $h-1$ $\underline{\text{height}}_{\text{tree}} > h$

- root node
- one AVL tree (left or right subtree) of height $h-1$
- one AVL tree of height either $(h-1)$ or $(h-2)$




$$\underline{n(h)} = 1 + n(h-1) + n(h-2)$$

lower bound needed

$$n(h) > n(h-1) + n(h-2)$$

and $\quad n(h-2) \leq n(h-1)$

$$n(h) > 2\, n(h-2)$$

$$> 2^2\, n(h-4)$$

$$\vdots$$

$$n(h) > 2^i\, n(h-2i)$$

Choose $i$ s.t $h - 2i = 2$

$$n(h) > 2^{h/2 - 1} n(2)$$

$$n(2) = 2$$

$$n(h) > 2^{h/2}$$

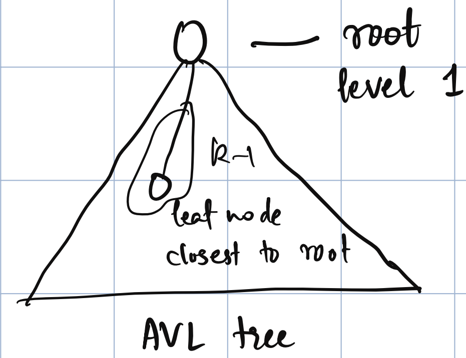
$$2 \log(n(h)) > h$$

or $\boxed{h = O(\log n(h))}$

The height of AVL tree is $O(\log n)$.

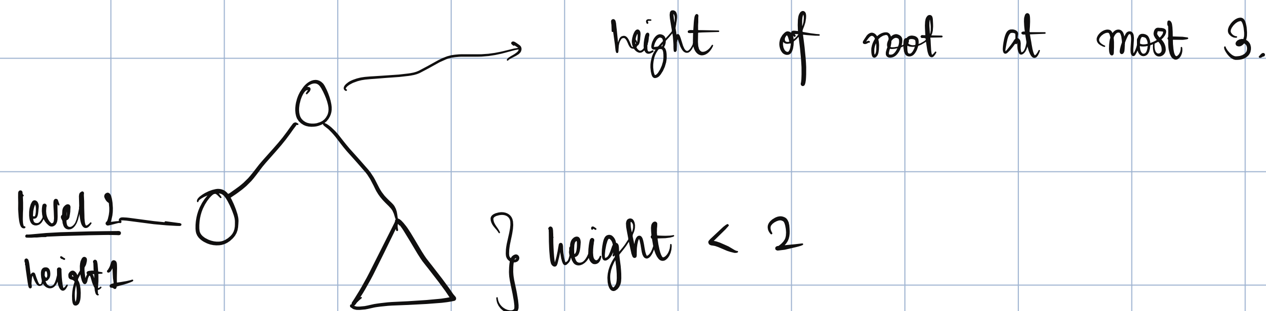
## Properties of AVL tree

Consider an AVL tree of $n$ nodes.

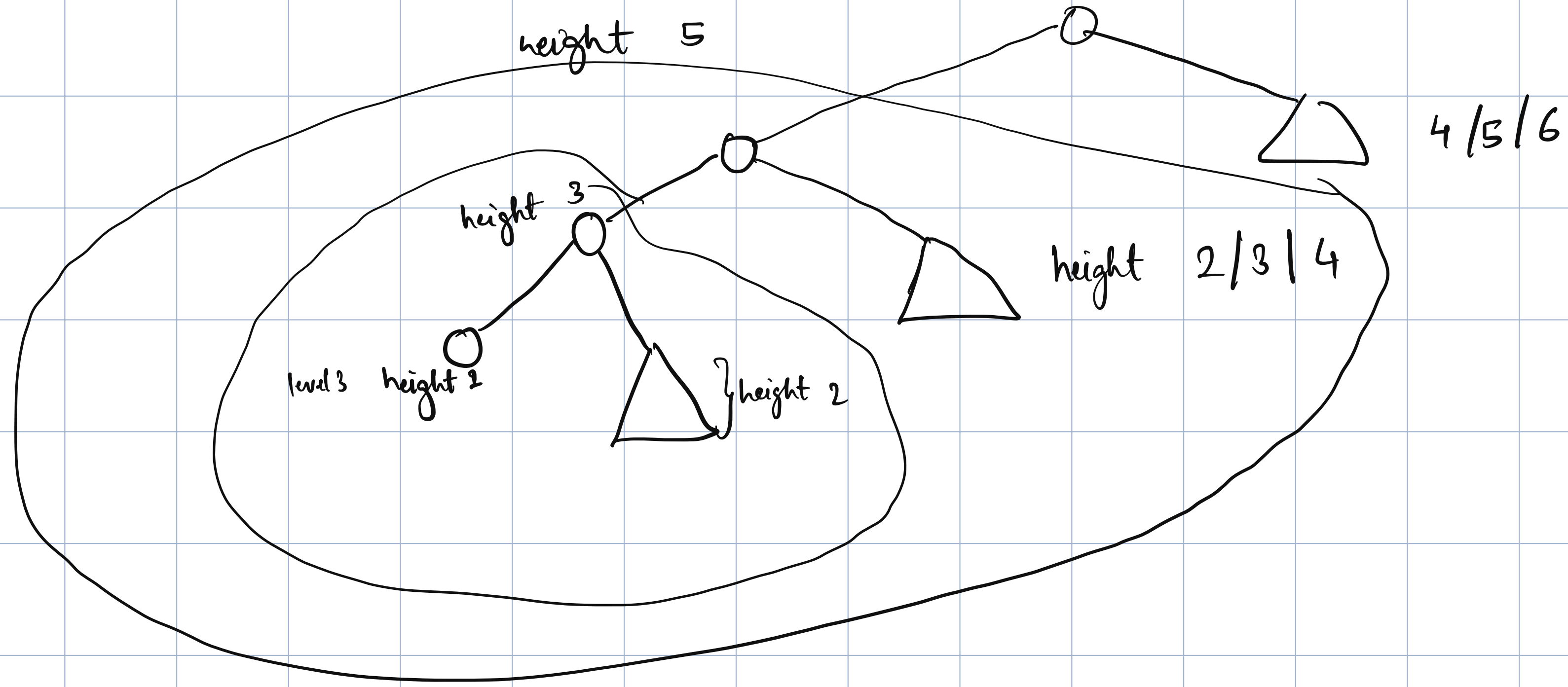



The height of AVL tree is $2k - 1$

Proof




$k = 2$ height 3




$k = 2 \qquad h \leq 3$

$k = 3 \qquad h \leq 5$

$k = 4 \qquad h \leq 7$

$$\boxed{h \leq 2k - 1}$$

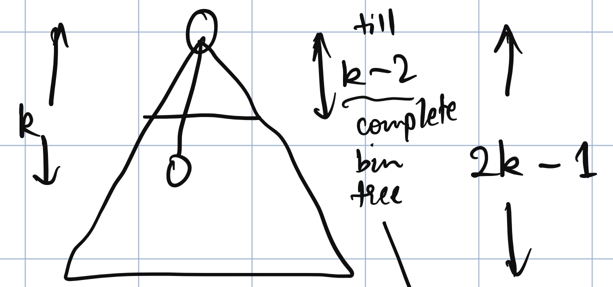
formal proof by induction.

## Proposition

If the closest leaf is at level k, then all nodes 1 … (k−2) have 2 children.

**Proof:**

In all proofs for AVL trees → use the property that $\text{diff}(h_1, h_2) < 1$



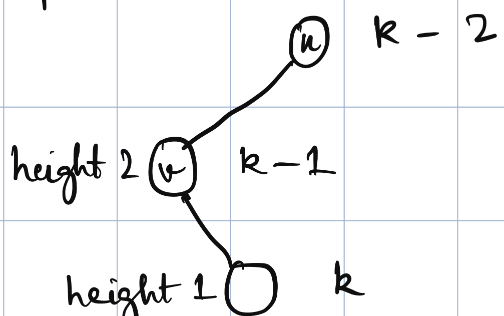

each internal node has exactly 2 children

all leaf nodes at same level

By contradiction.

Suppose $\exists\ u$ at level $(k-2)$ that has only one child $v$.

$v$ is at level $k-1$, so it can't be a leaf node




cannot be an AVL tree

BST: # of nodes can be linear in n

AVL:

$$\boxed{\begin{gathered} 2^{k-1} \le n \le 2^{2k-1} \\ 2^{(h-1/2)} \le n < 2^{h} \end{gathered}}$$

Height balance property is violated at u.

Contradiction

QED

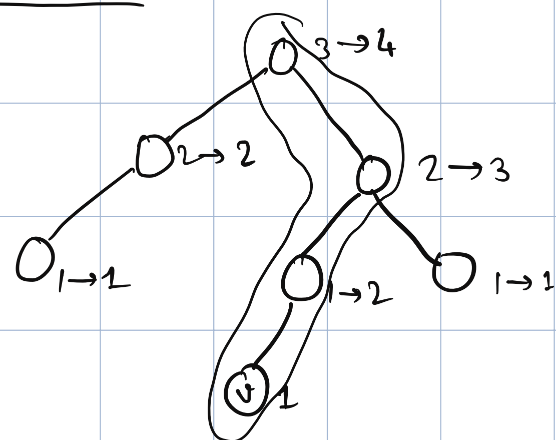
n is exponential in height

## Search, insertion and deletion in AVL tree

(i) Search - exactly BST

(ii) Insertion in AVL tree:

In BST → search, add node



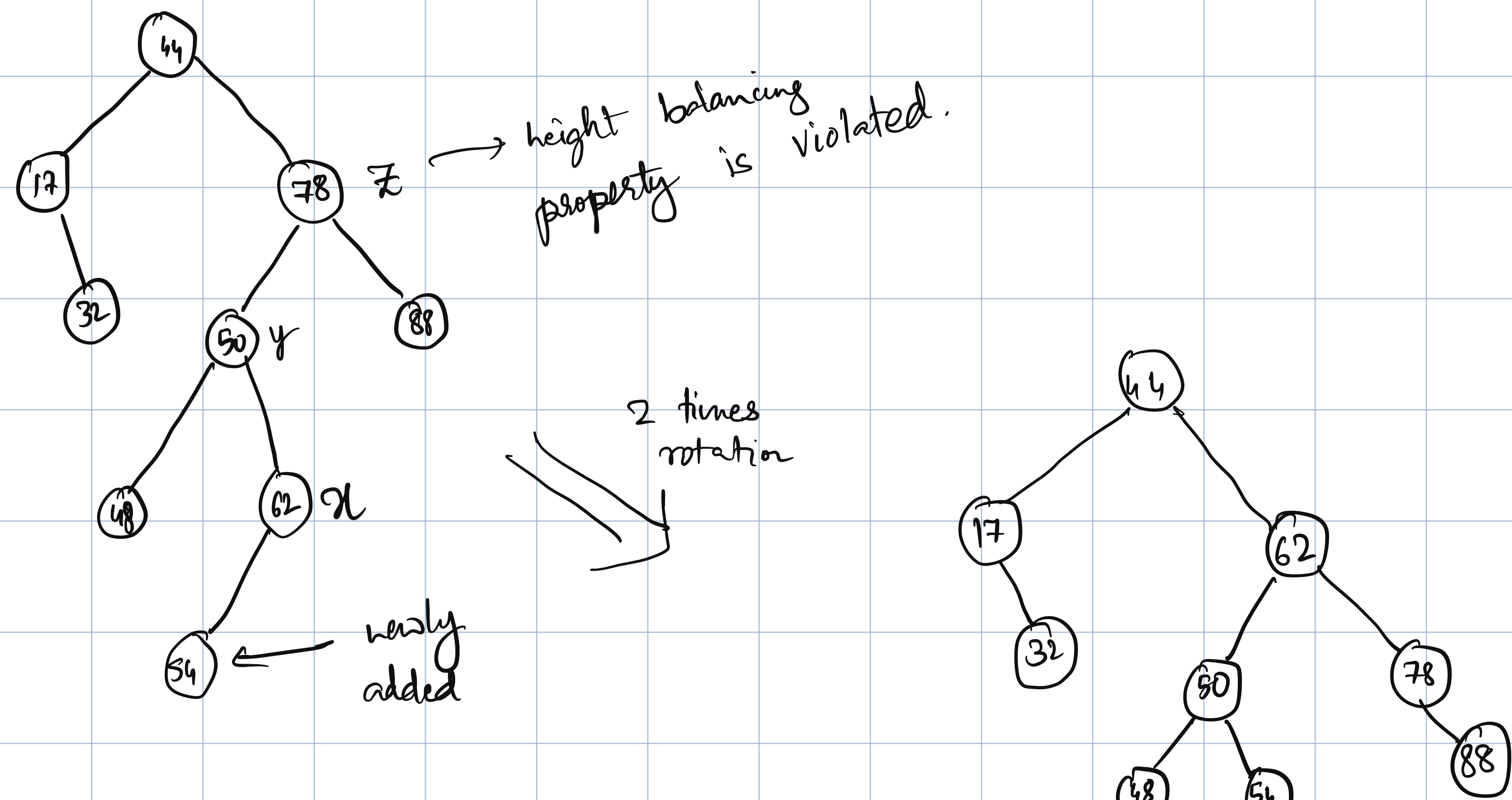

v: newly inserted node

* Inserting a node in AVL tree changes height of some nodes in T.

* Only those nodes that are ancestors of v.

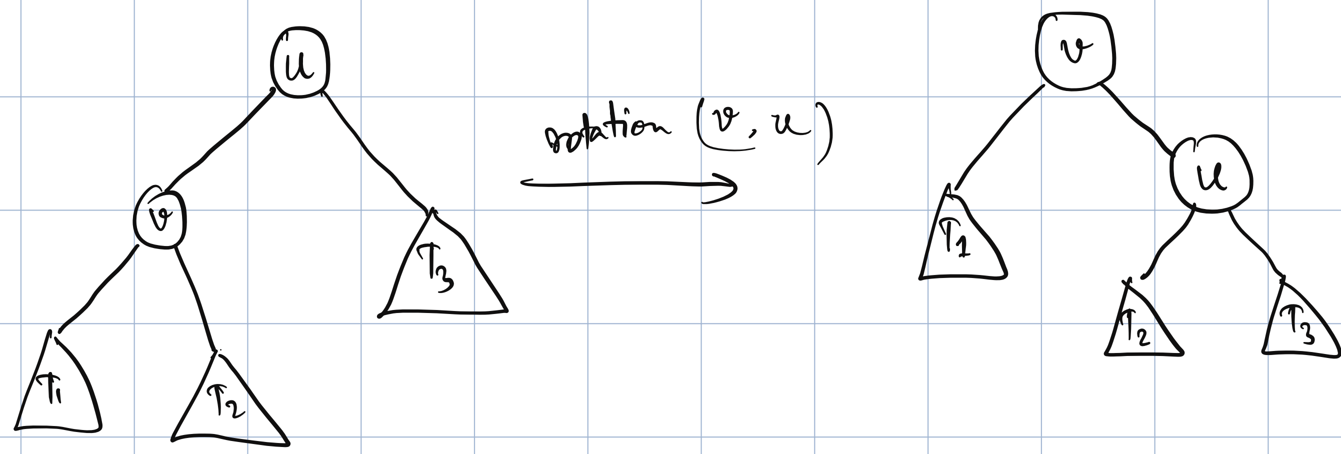
* Height balance property can violate at those nodes. (ancestors of v)

* Traverse up toward root from v until we find a a node where height balance property is violated (z)
  let z have child y and grandchild x
  on path
  v — z




**Rotation**: A process of locally reorganising BST

– u, v are two nodes s.t $u = \text{parent}(v)$



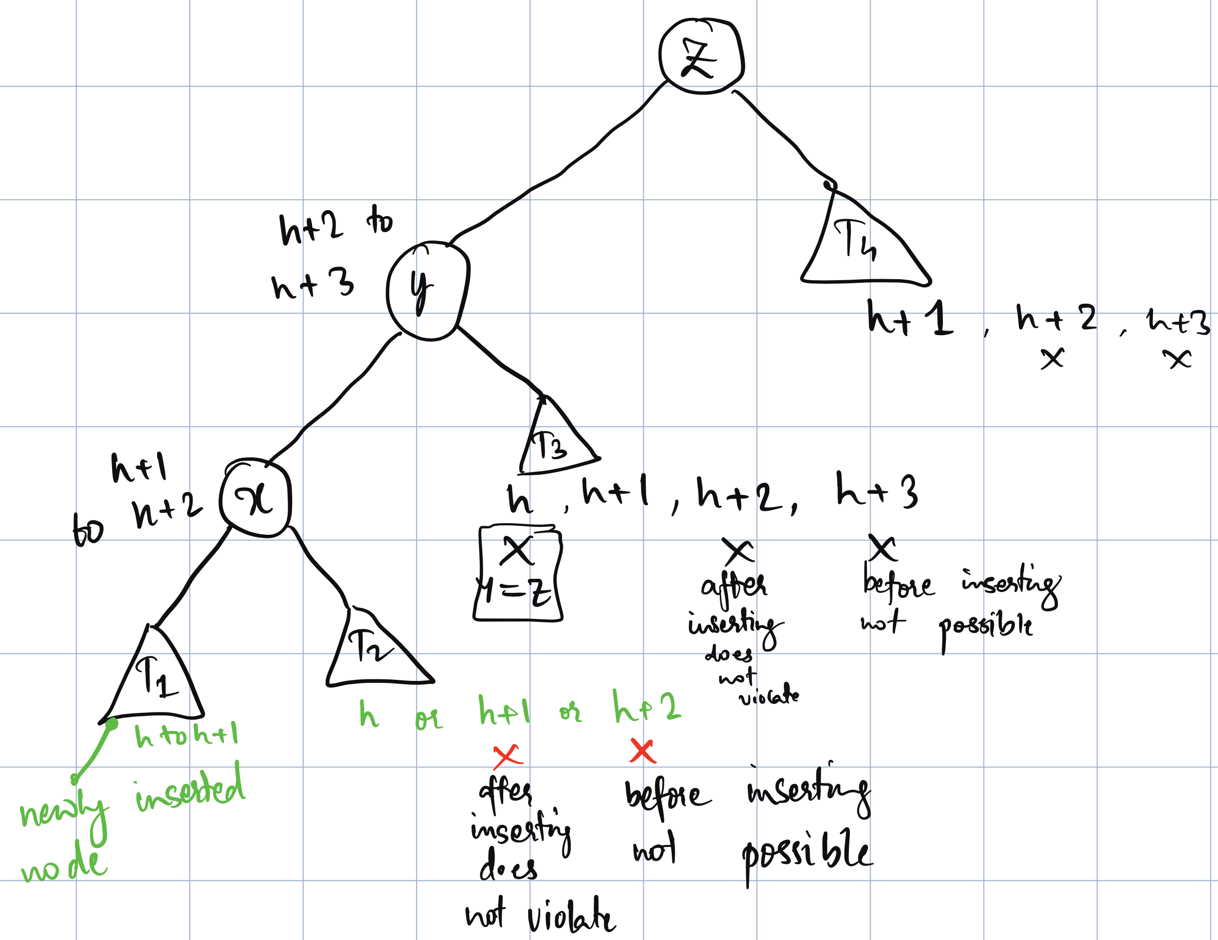

$$key(T_1) < key(v) < key(T_2) < key(u) < key(T_3)$$

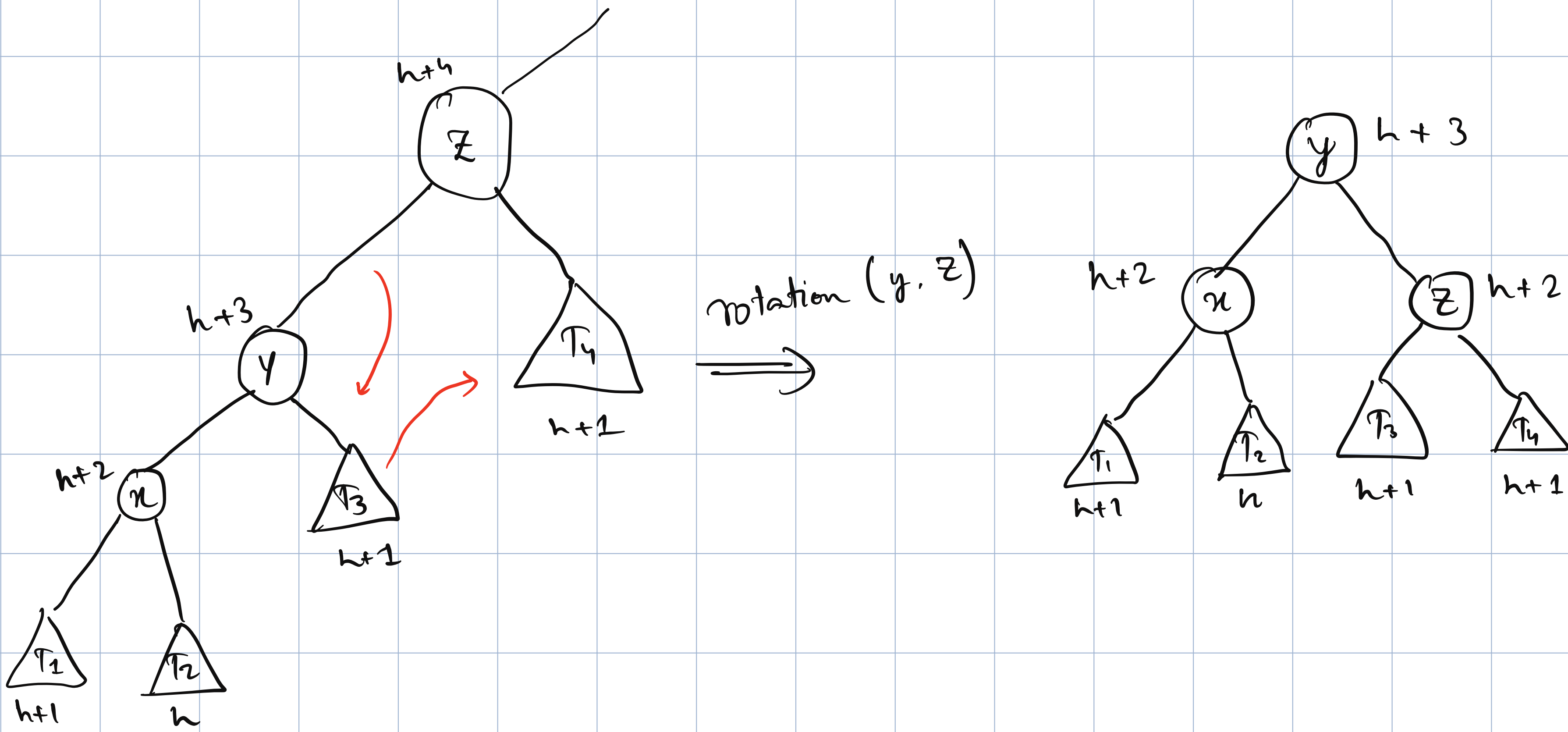
Insertion: 4 possible cases

y can be left / right child of z

x can be left / right child of y

case 1 Y is the left child of Z , X is the left subchild of Y




h+4
z
h+3
y
h+2
x
$T_1$
h+1
$T_2$
h
$T_3$
h+1
$T_4$
h+1
rotation (y, z)
y
h + 3
h+2
x
z
h + 2
$T_1$
h+1
$T_2$
h
$T_3$
h+1
$T_4$
h+ 1

**2024/09/12**

closest leaf node at $k$ $\Rightarrow$ height at most $2k-1$

from level 1 to $k$,

all internal nodes have 2 children

$z \to$ 1st node on which height balance is violated

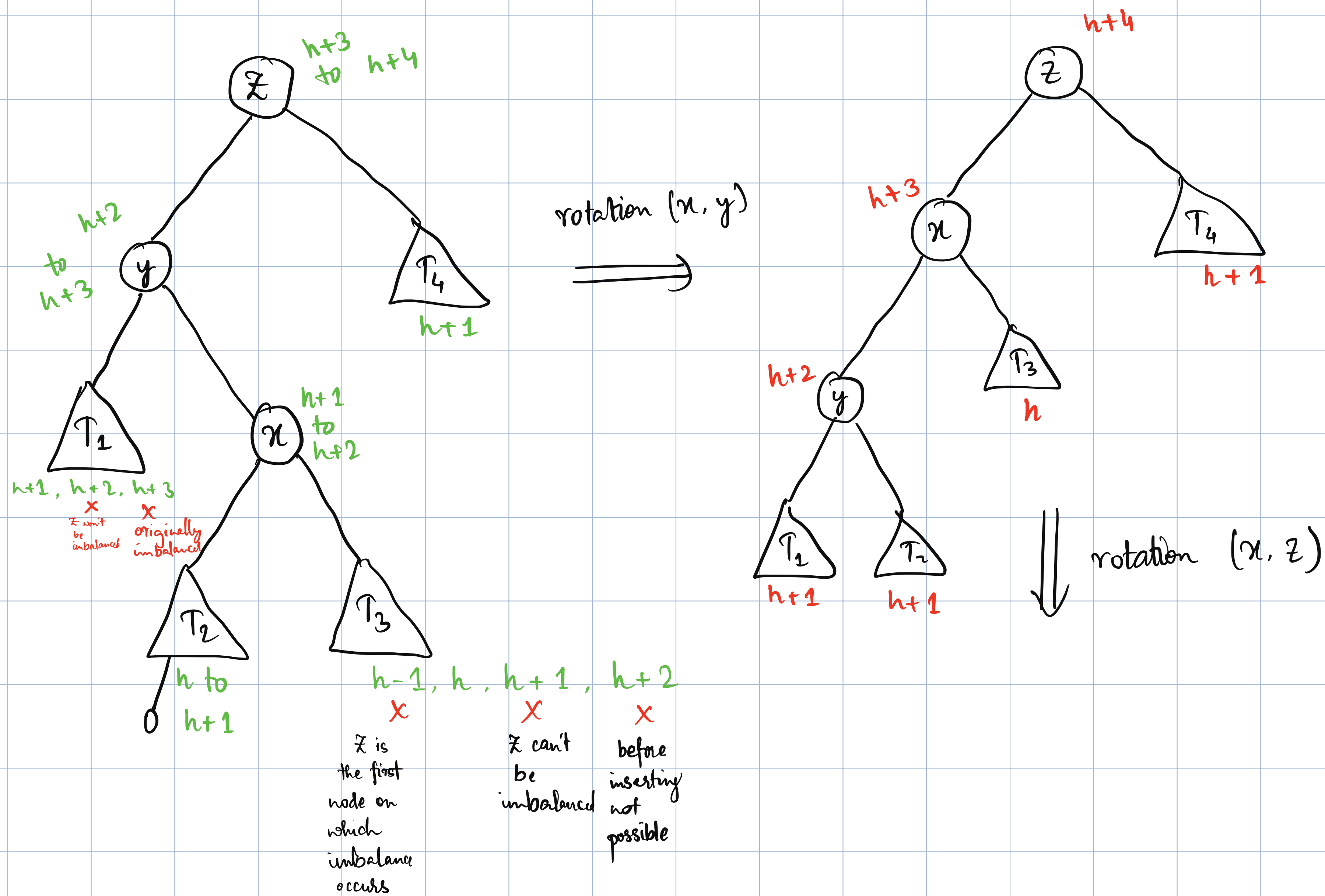
$y \to$ child of $z$ on the path to inserted node.

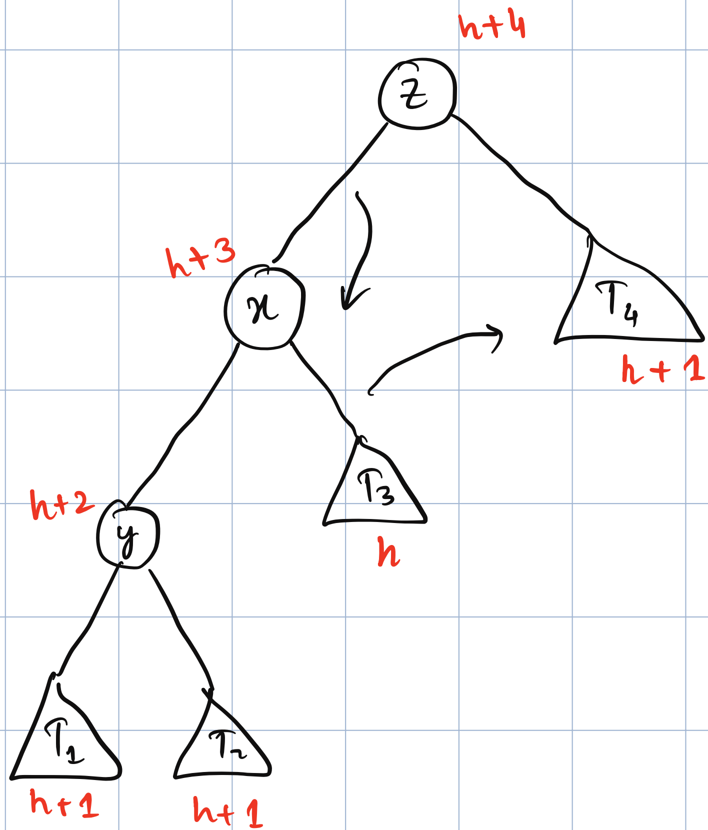
$x \to$ child of $y$ on the path to inserted node

Case (ii) $y \to$ right child of $z$

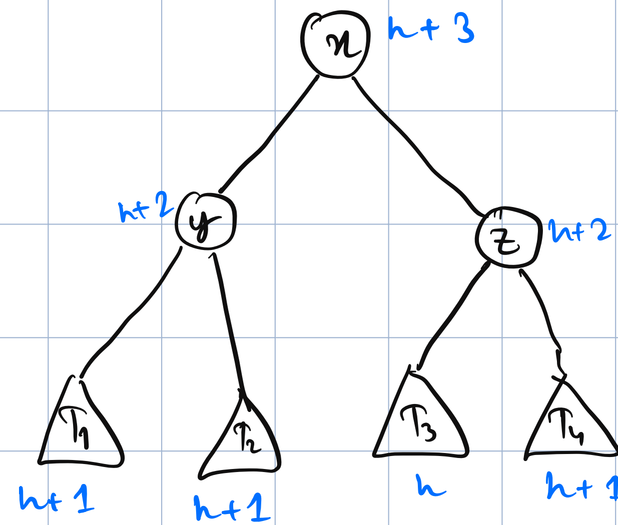
$x \to$ right child of $z$

analogous to case (i)

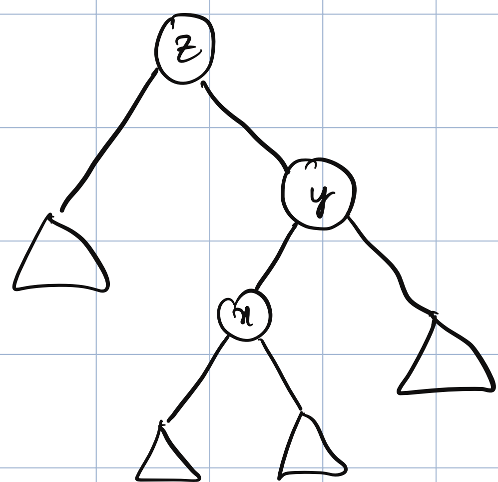

z
h+3 to h+4
y
h+2 to h+3
x
h+1 to h+2
T1
h+1, h+2, h+3
z won't be imbalanced
originally imbalanced
T2
h to h+1
0
T3
h-1, h, h+1, h+2
z is the first node on which imbalance occurs
z can't be imbalanced
before inserting not possible
T4
h+1
rotation (x, y)
z
h+4
x
h+3
y
h+2
T1
h+1
T2
h+1
T3
h
T4
h+1
rotation (x, z)




rotation $(x, z)$ $\Longrightarrow$




Case (iv)




identical to case (iii)

Time complexity –

1. insesting like BST $O(\log n)$
2. identifying $x$, $y$ and $z$ $O(\log n)$
3. rotation (single / double) $O(1)$

Deletion in

* Let $W$ be the deleted node.
* Let $Z$ be the first imbalanced node in the path from $W$ to root.

  Let $y$ be the child of $Z$ with larger height.
* Let $x$ be the child of $y$ with larger height

Y can be left/right child of Z

X can be left/right child of y

Case (i)

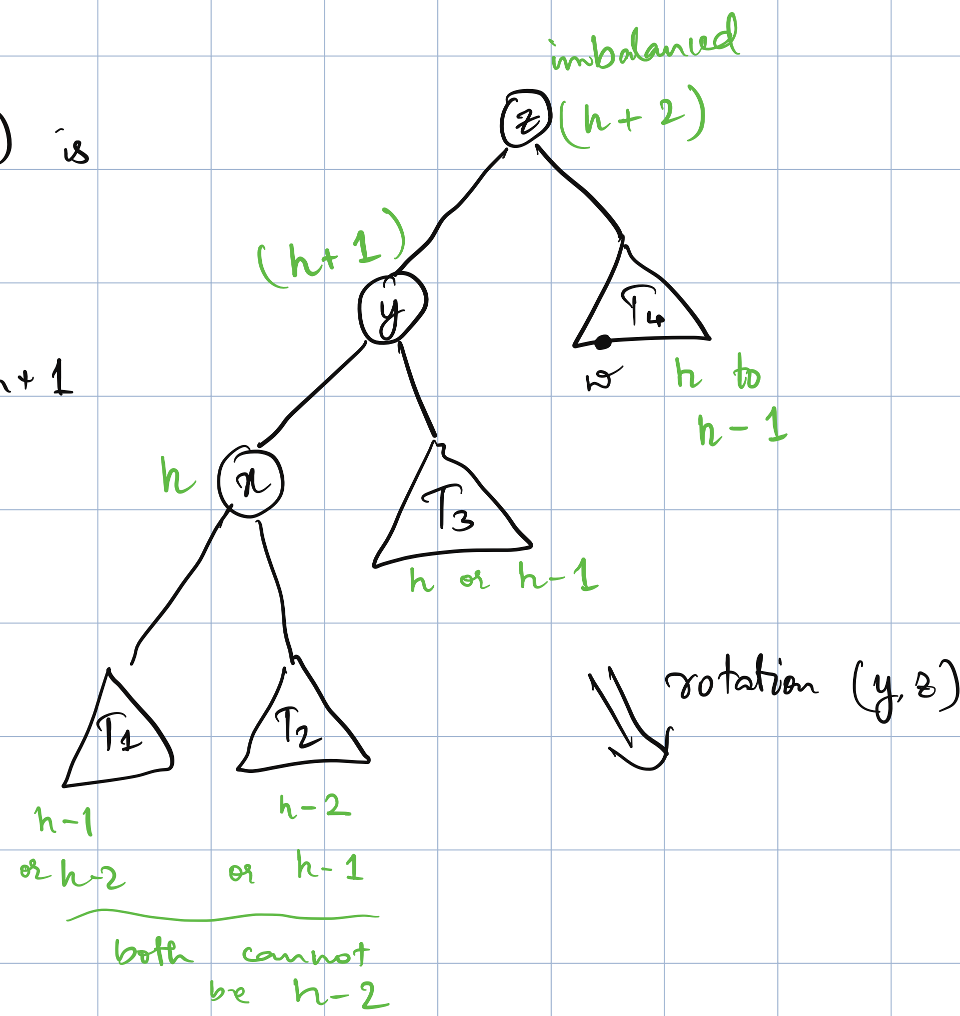
Deletion happens at $T_4$, $h(T_4)$ is reduced from h to h − 1.

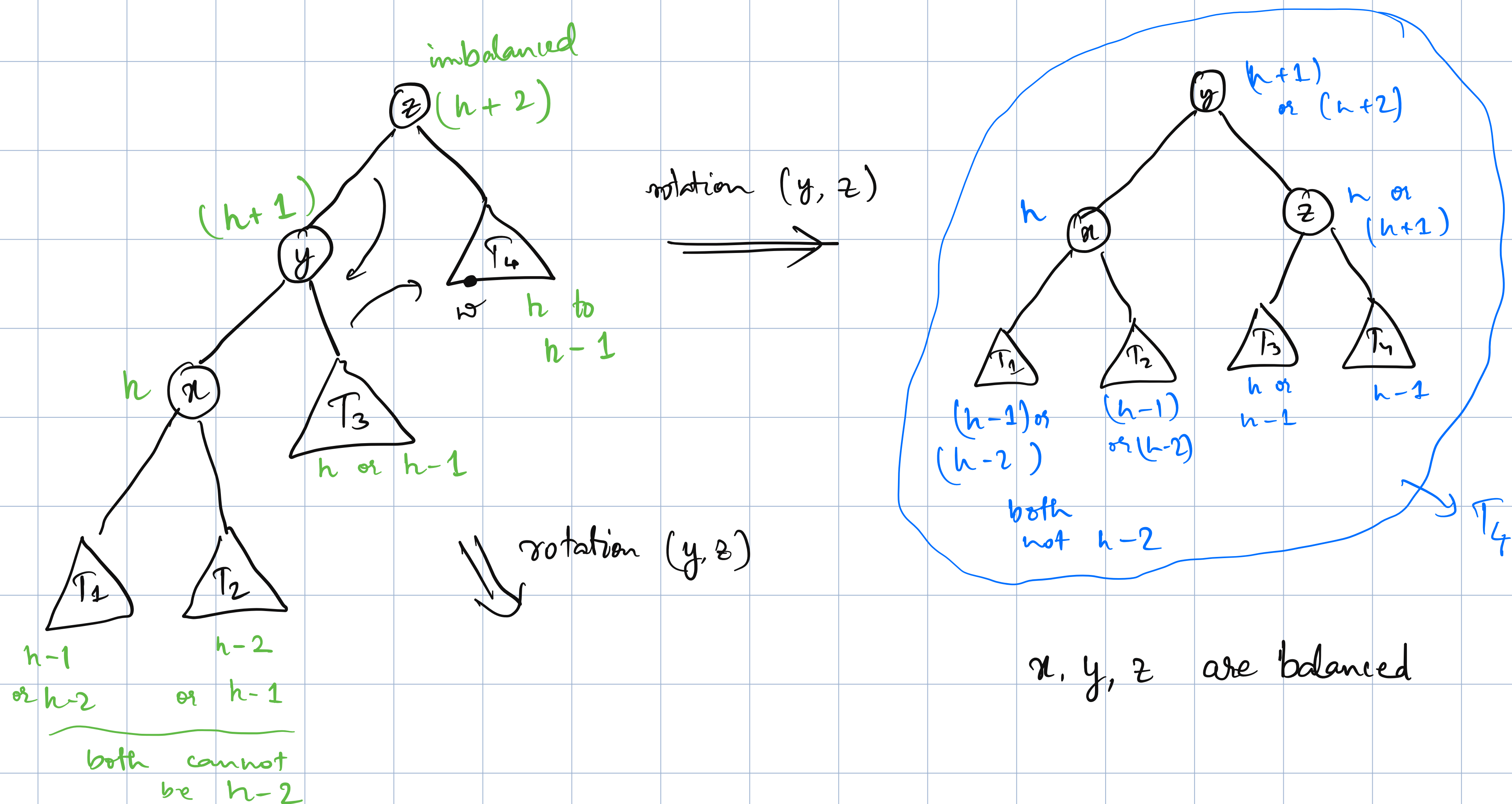
Z is imbalanced $\Rightarrow$ $h_T(y) = h+1$

x has larger height than $T_3$

so $h_T(x) = h$

Since $h_T(T_1)$, $h_T(T_2)$ can be either $(h-1)$ or $(h-2)$



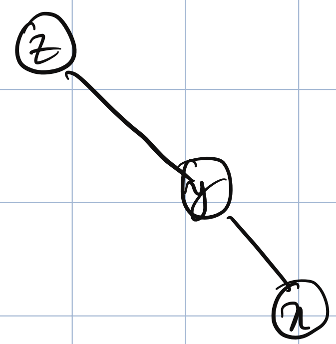



after rotation $h_t(T)$ can be reduced. So need to continue up further.

repeat the algorithm considering T as new $T_4$.

* In the worst case repitition needs to be done $O(\log n)$ times.

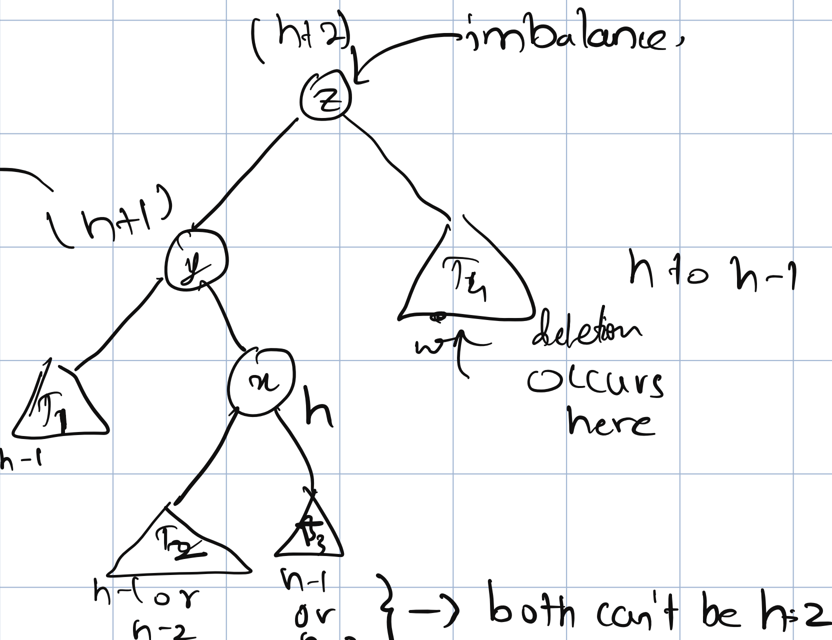
Case (ii)




Analogous to Case i

Case (iii)

assume h+1 so as to let imbalance be there.




→ both can't be h-2 as ht. of x is h.

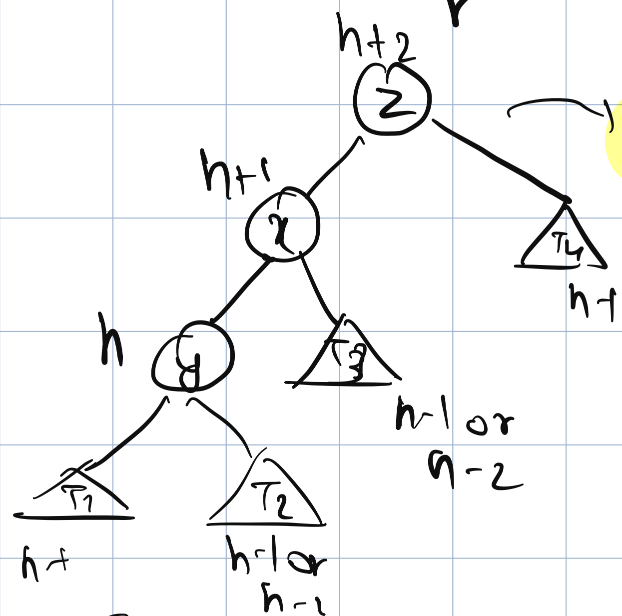
As before ht(y) = h+1

since y is balanced ht(x) = h

ht.($T_1$) is h or h-1

if ht($T_1$) is h then we would have picked root of $T_1$ as x

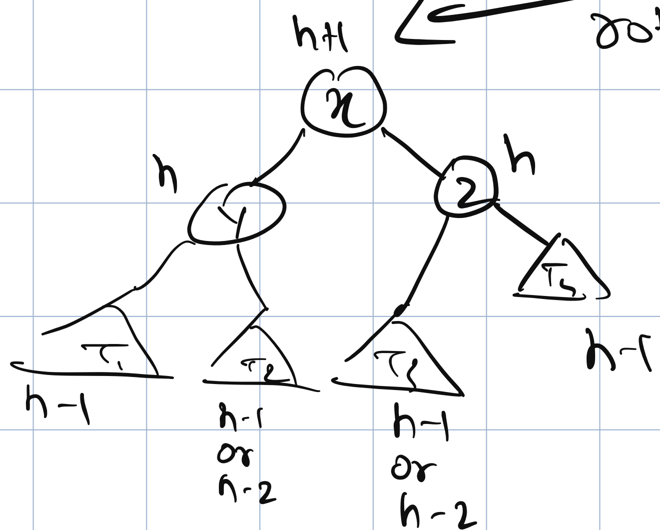
rotation (x, y)




imbalance is here but

Can imbalance here due to ht($T_3$) = h-2

rotate (x, z)

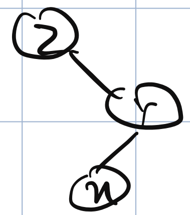



$:= T$

Anup

ht of T reduces from h+2 to h+1, repeat considering T as new $T_4$

Case iv :




analogous to previous case (case iii).

Time complexity :-

- Deletion using BST O(log n)
- In worst case, we need to go up to root - O(log n)
  - In each we need to perform O(1) rotation.