balanced trees 2

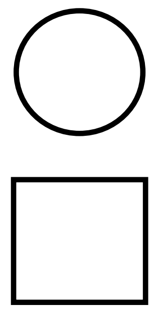
AVL trees

basic definitions

two kinds of nodes:

- *inner nodes/implementation* nodes forming a binary tree.
- *ghost nodes* only for accounting. Added such that each inner node has two sons.

symbols

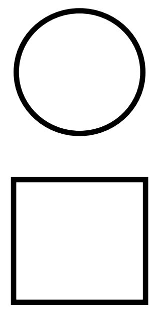


basic definitions

two kinds of nodes:

- *inner nodes/implementation* nodes forming a binary tree.
- *ghost nodes* only for accounting. Added such that each inner node has two sons.

symbols



size and height

$$|T| = \#\text{ inner nodes}$$

$$nl(T) = \#\text{ leaves} = \#\text{ ghost nodes}$$

$$h(T): \text{ ghost nodes counted}$$

basic definitions

two kinds of nodes:

- *inner nodes/implementation* nodes forming a binary tree.
- *ghost nodes* only for accounting. Added such that each inner node has two sons.

symbols

○

□

size and height

$$|T| = \#\,\text{inner nodes}$$

$$nl(T) = \#\,\text{leaves} = \#\,\text{ghost nodes}$$

$$h(T):\ \text{ghost nodes counted}$$

Lemma 1.

$$nl(T) = |T| + 1$$

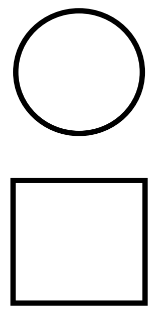
Proof. induction on $|T|$:

basic definitions

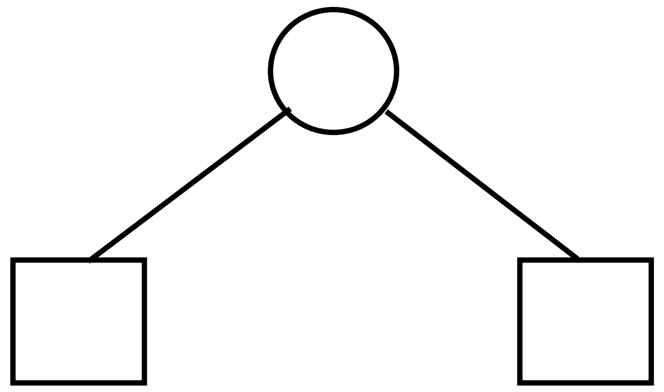
two kinds of nodes:

- *inner nodes/implementation* nodes forming a binary tree.
- *ghost nodes* only for accounting. Added such that each inner node has two sons.

symbols



$|T| = 1$



size and height

$$|T| = \#\text{ inner nodes}$$

$$nl(T) = \#\text{ leaves} = \#\text{ ghost nodes}$$

$$h(T): \text{ ghost nodes counted}$$

Lemma 1.

$$nl(T) = |T| + 1$$

Proof. induction on $|T|$:

basic definitions

two kinds of nodes:

- *inner nodes/implementation* nodes forming a binary tree.
- *ghost nodes* only for accounting. Added such that each inner node has two sons.

size and height

$$|T| = \#\text{ inner nodes}$$

$$nl(T) = \#\text{ leaves} = \#\text{ ghost nodes}$$

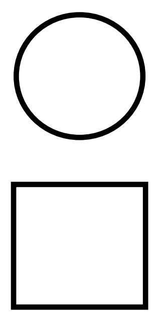
$$h(T):\ \text{ghost nodes counted}$$

Lemma 1.

$$nl(T) = |T| + 1$$

Proof. induction on $|T|$:

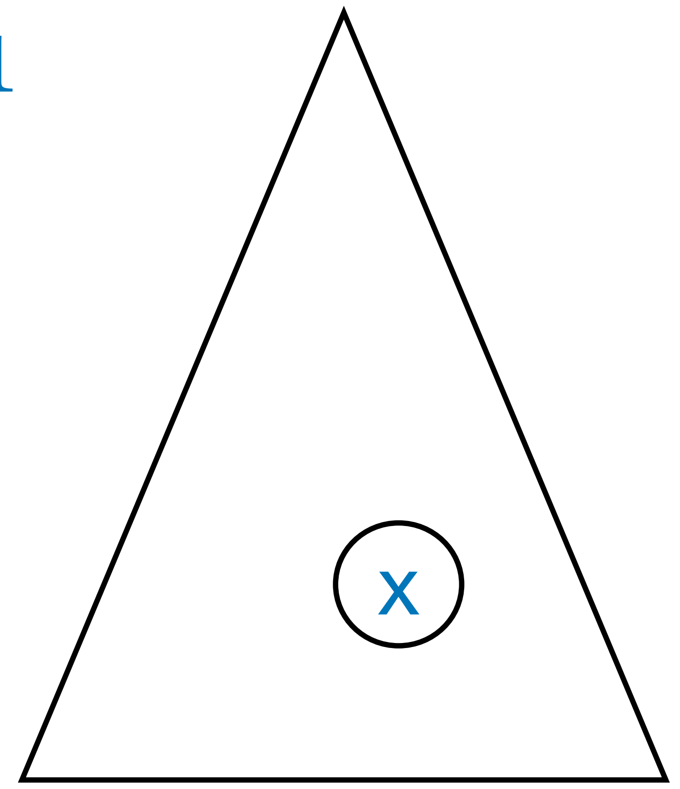
symbols



$|T| = 1$

$n \to n + 1$

$|T| = n$



$x \in T$ *interior node*

basic definitions

two kinds of nodes:

- *inner nodes/implementation* nodes forming a binary tree.
- *ghost nodes* only for accounting. Added such that each inner node has two sons.

size and height

$$|T| = \#\text{ inner nodes}$$

$$nl(T) = \#\text{ leaves} = \#\text{ ghost nodes}$$

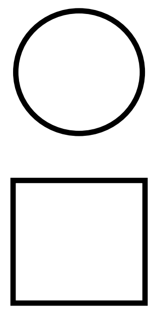
$$h(T):\ \text{ghost nodes counted}$$

Lemma 1.

$$nl(T) = |T| + 1$$

Proof. induction on $|T|$:

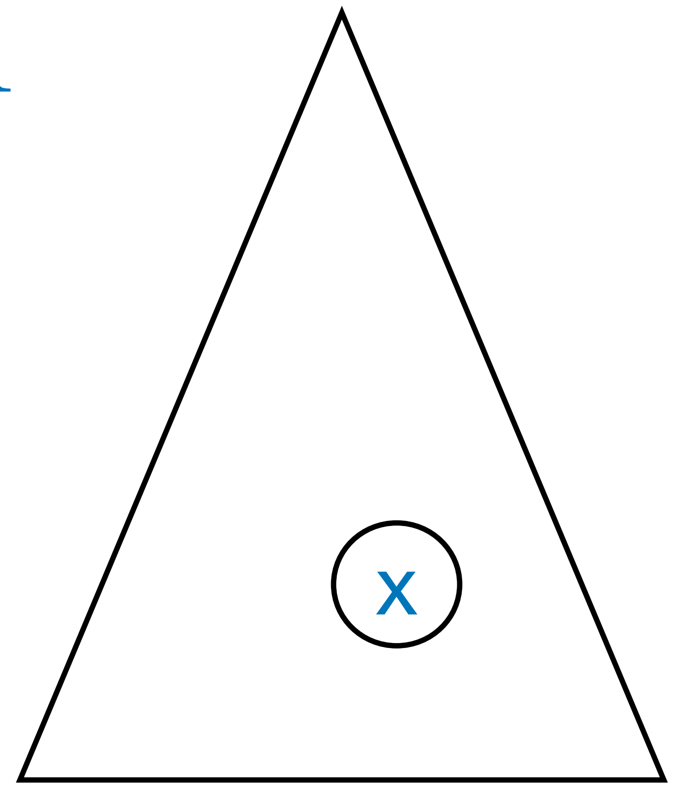
symbols

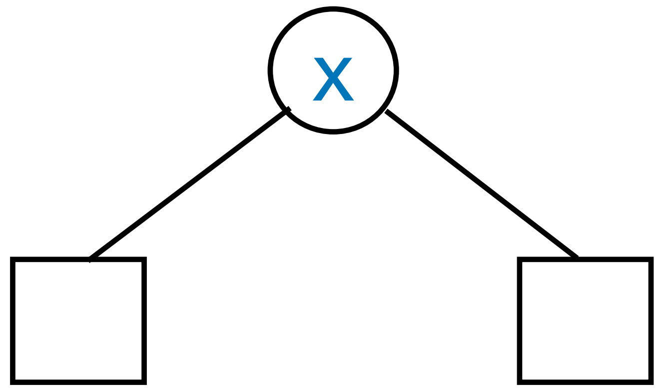


$|T| = 1$

$n \to n+1$

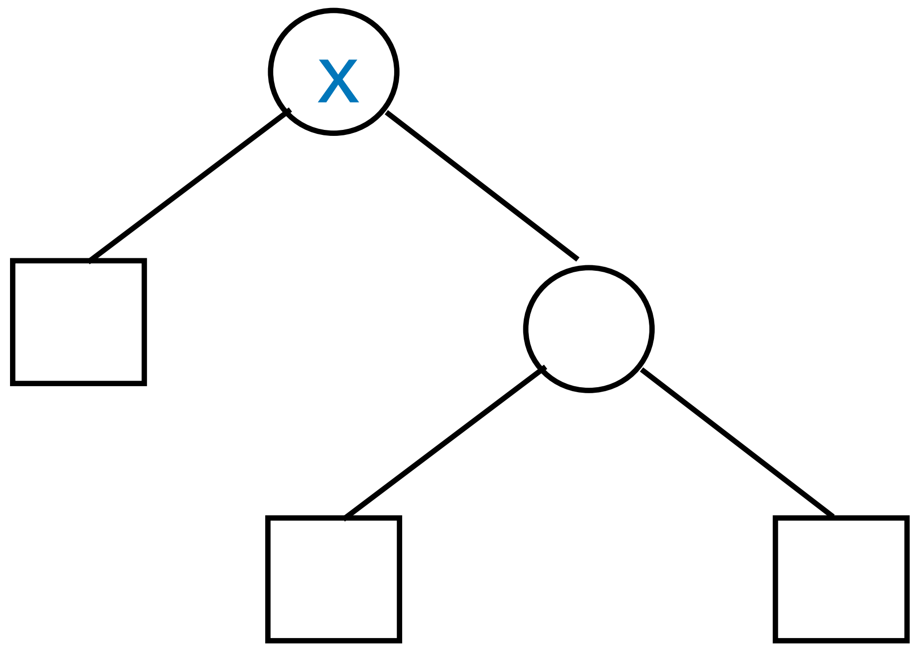
$|T| = n$





$x \in T$ *interior node*

- with no interior son



basic definitions

two kinds of nodes:

- *inner nodes/implementation* nodes forming a binary tree.
- *ghost nodes* only for accounting. Added such that each inner node has two sons.

size and height

$$|T| = \#\text{ inner nodes}$$

$$nl(T) = \#\text{ leaves} = \#\text{ ghost nodes}$$

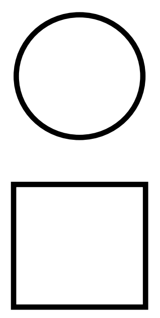
$$h(T):\ \text{ghost nodes counted}$$

Lemma 1.

$$nl(T) = |T| + 1$$

Proof. induction on $|T|$:

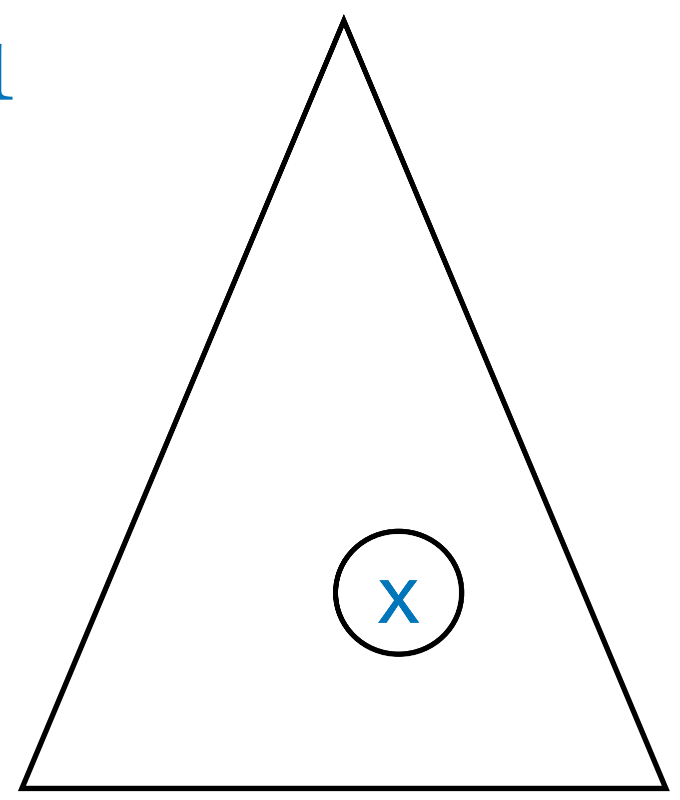
symbols

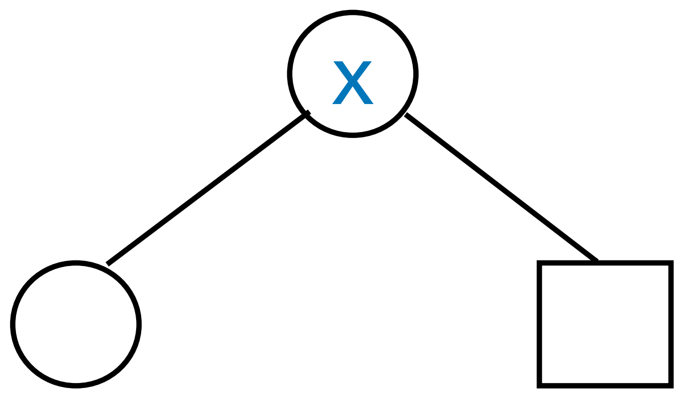


$|T| = 1$

$n \to n + 1$

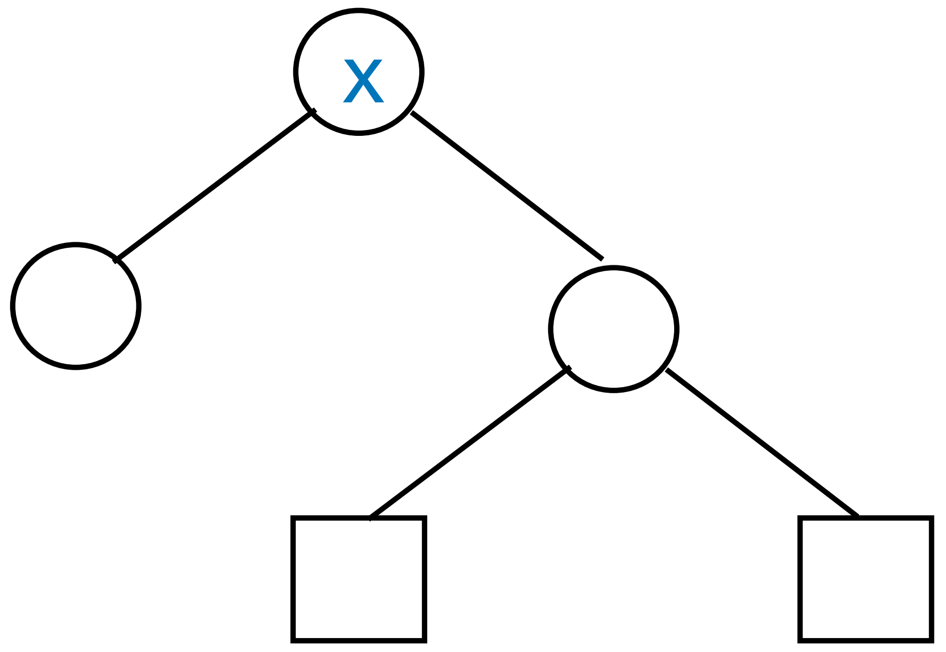
$|T| = n$





$x \in T$ *interior node*

- with one interior son

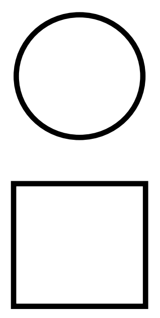


basic definitions

two kinds of nodes:

- *inner nodes/implementation* nodes forming a binary tree.
- *ghost nodes* only for accounting. Added such that each inner node has two sons.

symbols



size and height

$$|T| = \#\text{ inner nodes}$$

$$nl(T) = \#\text{ leaves} = \#\text{ ghost nodes}$$

$$h(T): \text{ ghost nodes counted}$$

Lemma 1.

$$nl(T) = |T| + 1$$

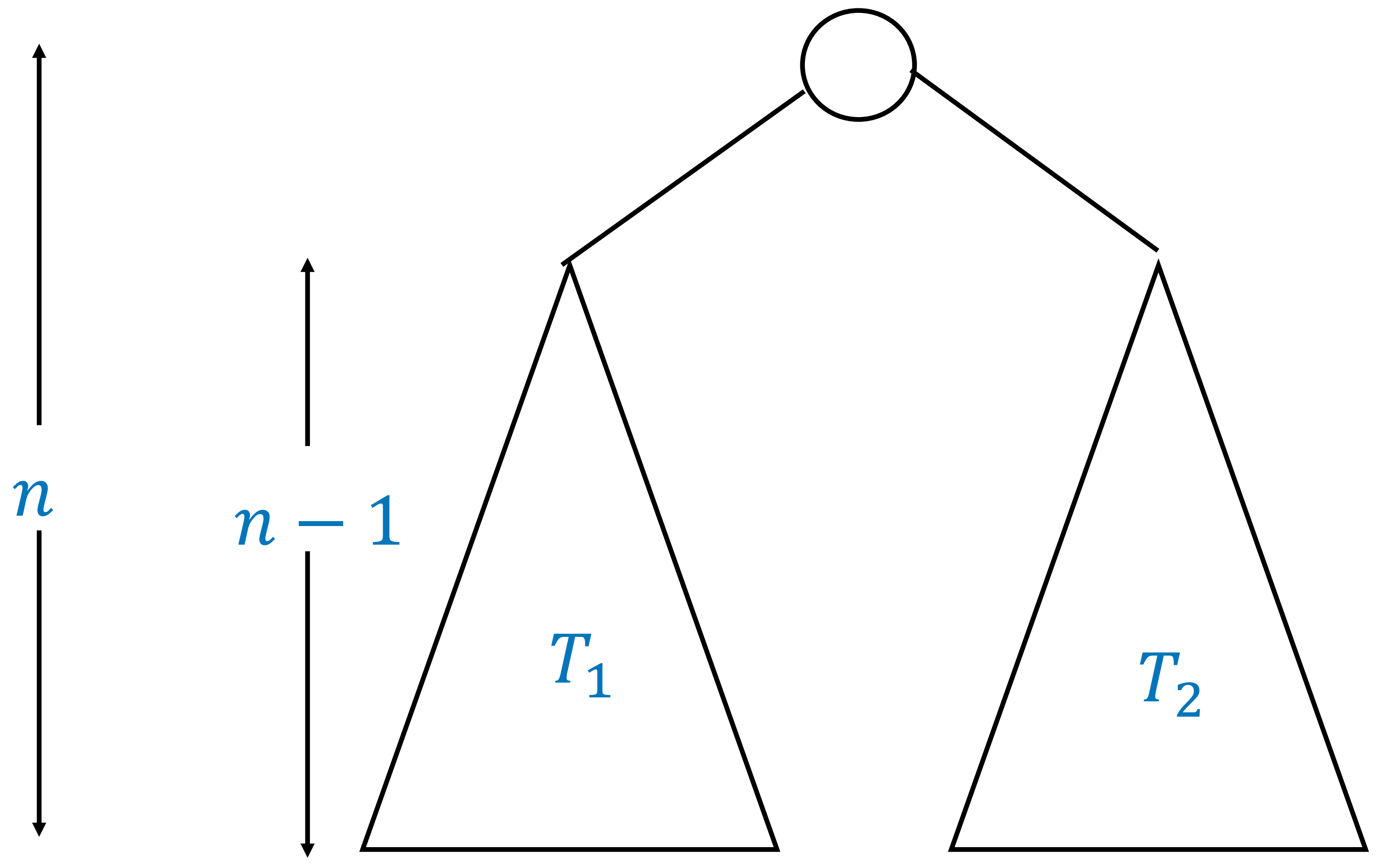
Proof. induction on $|T|$:

balance of a node $x \in T$

$$bal(x) = h(L(x)) - h(R(x))$$

AVL-property:

$$AVL(T) \equiv \forall x \in T : bal(x) \in \{-1, 0, 1\}$$

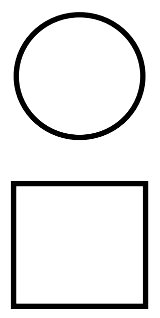


basic definitions

two kinds of nodes:

- *inner nodes/implementation* nodes forming a binary tree.
- *ghost nodes* only for accounting. Added such that each inner node has two sons.

symbols



size and height

$$|T| = \#\text{ inner nodes}$$

$$nl(T) = \#\text{ leaves} = \#\text{ ghost nodes}$$

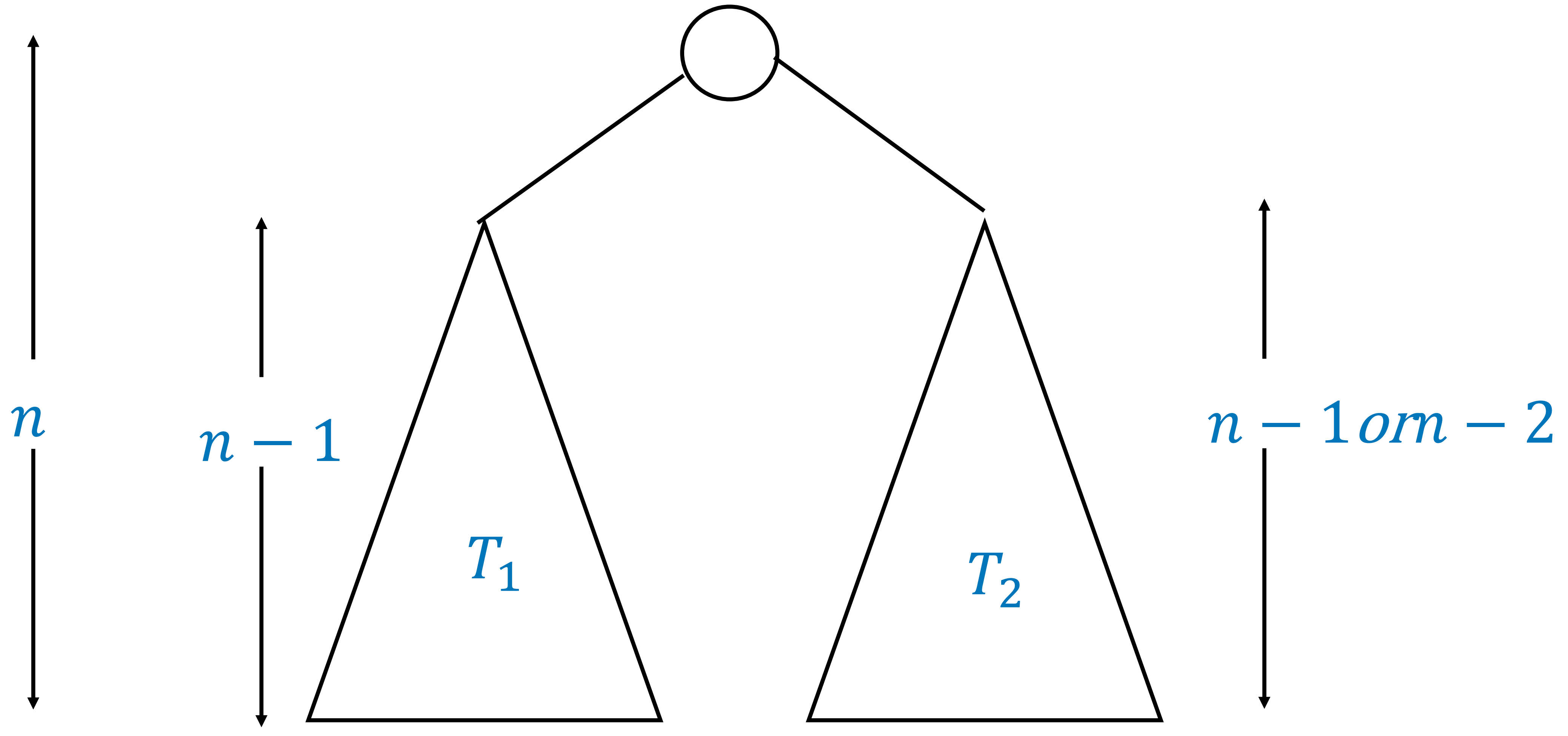
$$h(T):\ \text{ghost nodes counted}$$

balance of a node $x \in T$

$$bal(x) = h(L(x)) - h(R(x))$$

AVL-property:

$$AVL(T) \equiv \forall x \in T : bal(x) \in \{-1, 0, 1\}$$



Lemma 1.

$$nl(T) = |T| + 1$$

Proof. induction on $|T|$:

size versus depth

Lemma 2. *AVL trees are balanced.*

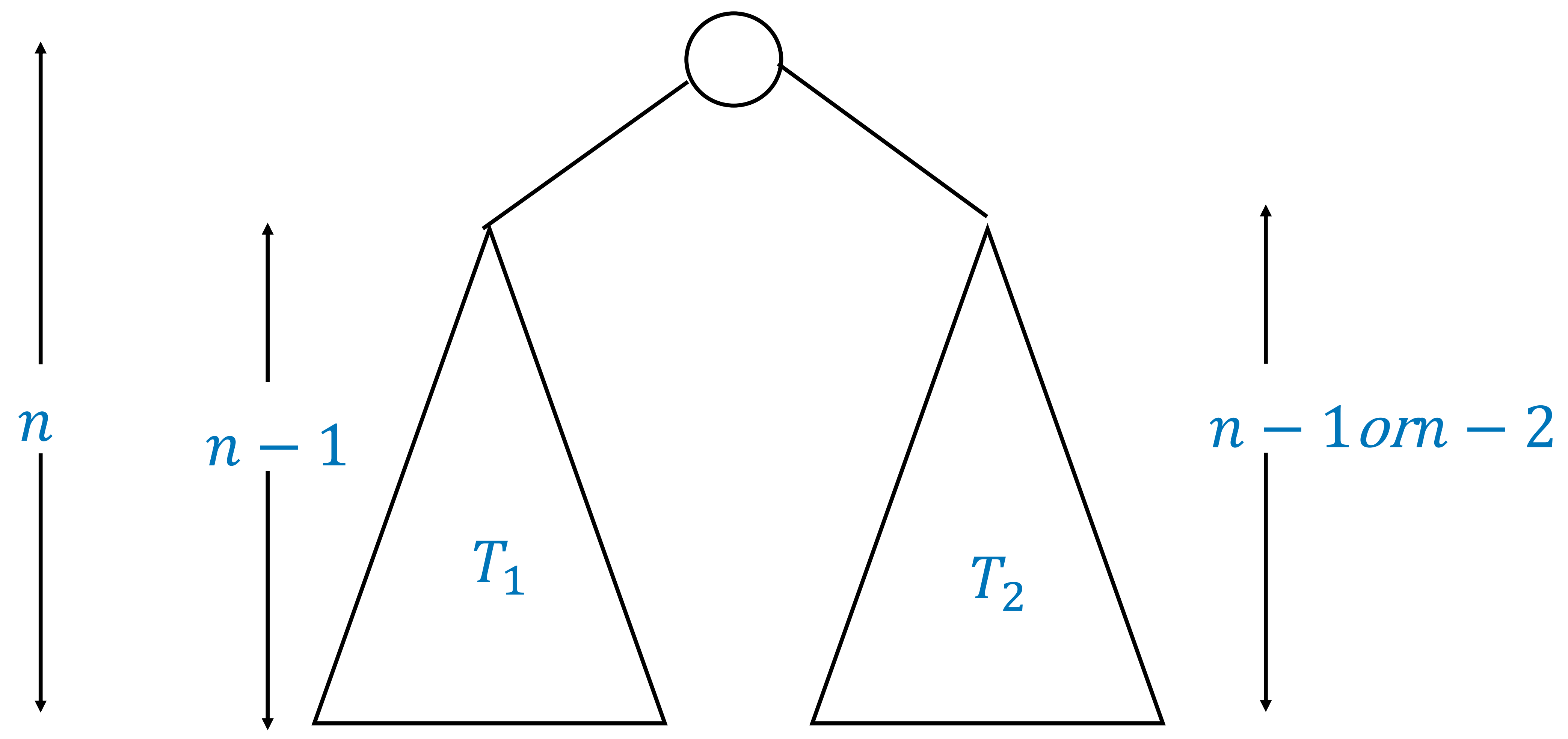
$$|T| = n \wedge AVL(T) \quad \rightarrow \quad h(T) = O(\log n)$$

balance of a node $x \in T$

$$bal(x) = h(L(x)) - h(R(x))$$

AVL-property:

$$AVL(T) \equiv \forall x \in T : bal(x) \in \{-1, 0, 1\}$$



excursion

Lemma 2. *AVL trees are balanced.*

$$|T| = n \wedge AVL(T) \quad \rightarrow \quad h(T) = O(\log n)$$

Def: *Fibonacci numbers*

$$\begin{aligned} F_0 &= 0 \\ F_1 &= 1 \\ F_n &= F_{n-1} + F_{n-2} \end{aligned}$$

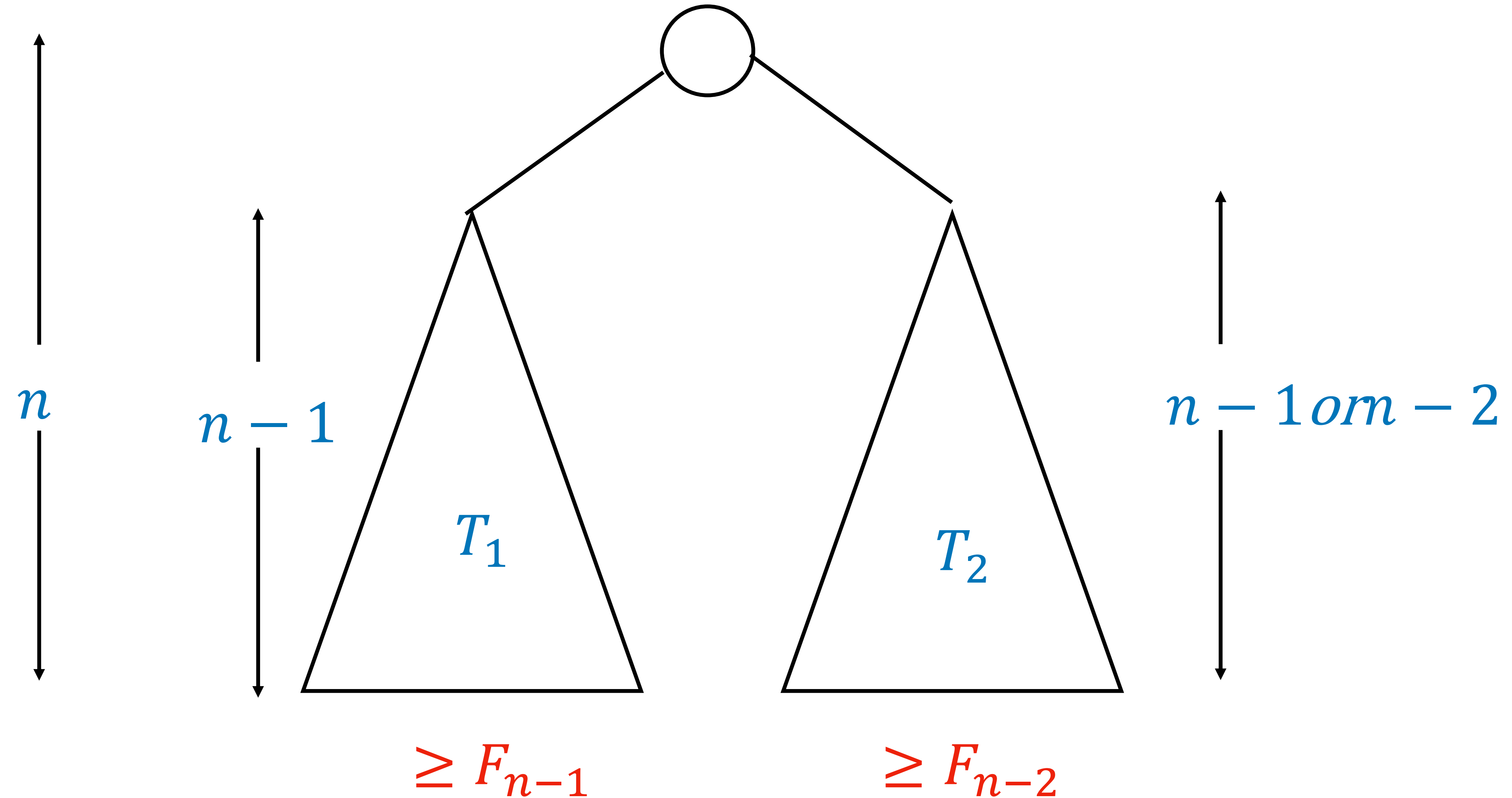
$$\boxed{h(T) = n \rightarrow nl(T) = |T| + 1 \geq F_n}$$

balance of a node $x \in T$

$$bal(x) = h(L(x)) - h(R(x))$$

AVL-property:

$$AVL(T) \equiv \forall x \in T : bal(x) \in \{-1, 0, 1\}$$



excursion

Lemma 2. *AVL trees are balanced.*

$$|T| = n \wedge AVL(T) \quad \rightarrow \quad h(T) = O(\log n)$$

Def: *Fibonacci numbers*

$$\begin{aligned} F_0 &= 0 \\ F_1 &= 1 \\ F_n &= F_{n-1} + F_{n-2} \end{aligned}$$

$$\boxed{h(T) = n \rightarrow nl(T) = |T| + 1 \geq F_n}$$

Lemma 3. *Moivre-Binet formula*

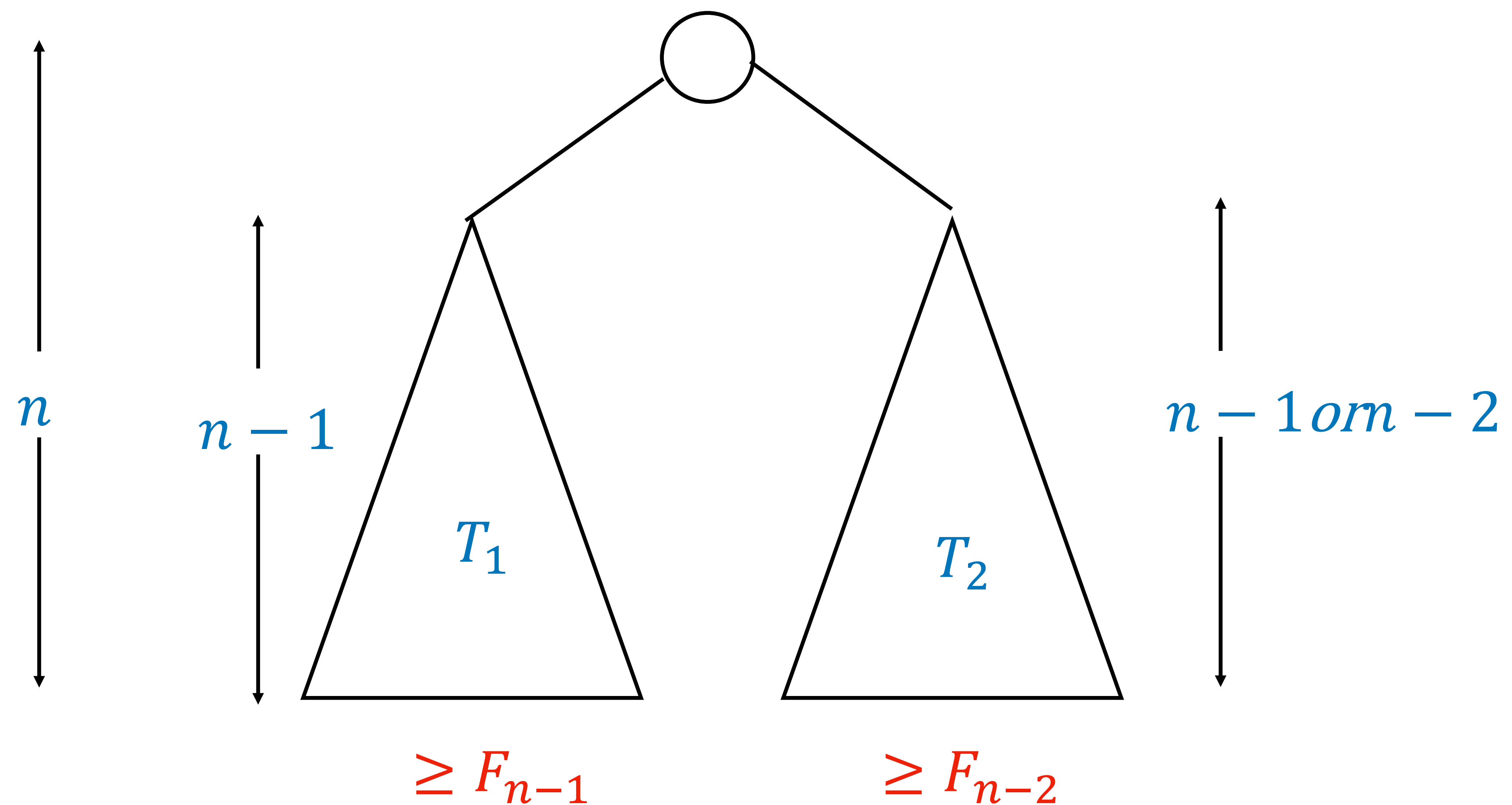
$$F_n = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right)$$

balance of a node $x \in T$

$$bal(x) = h(L(x)) - h(R(x))$$

AVL-property:

$$AVL(T) \equiv \forall x \in T : bal(x) \in \{-1, 0, 1\}$$



Lemma 2. *AVL trees are balanced.*

$$|T| = n \wedge AVL(T) \quad \rightarrow \quad h(T) = O(\log n)$$

Def: *Fibonacci numbers*

$$\begin{aligned} F_0 &= 0 \\ F_1 &= 1 \\ F_n &= F_{n-1} + F_{n-2} \end{aligned}$$

$$\boxed{h(T) = n \rightarrow nl(T) = |T| + 1 \geq F_n}$$

Lemma 3. *Moivre-Binet formula*

$$F_n = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right)$$

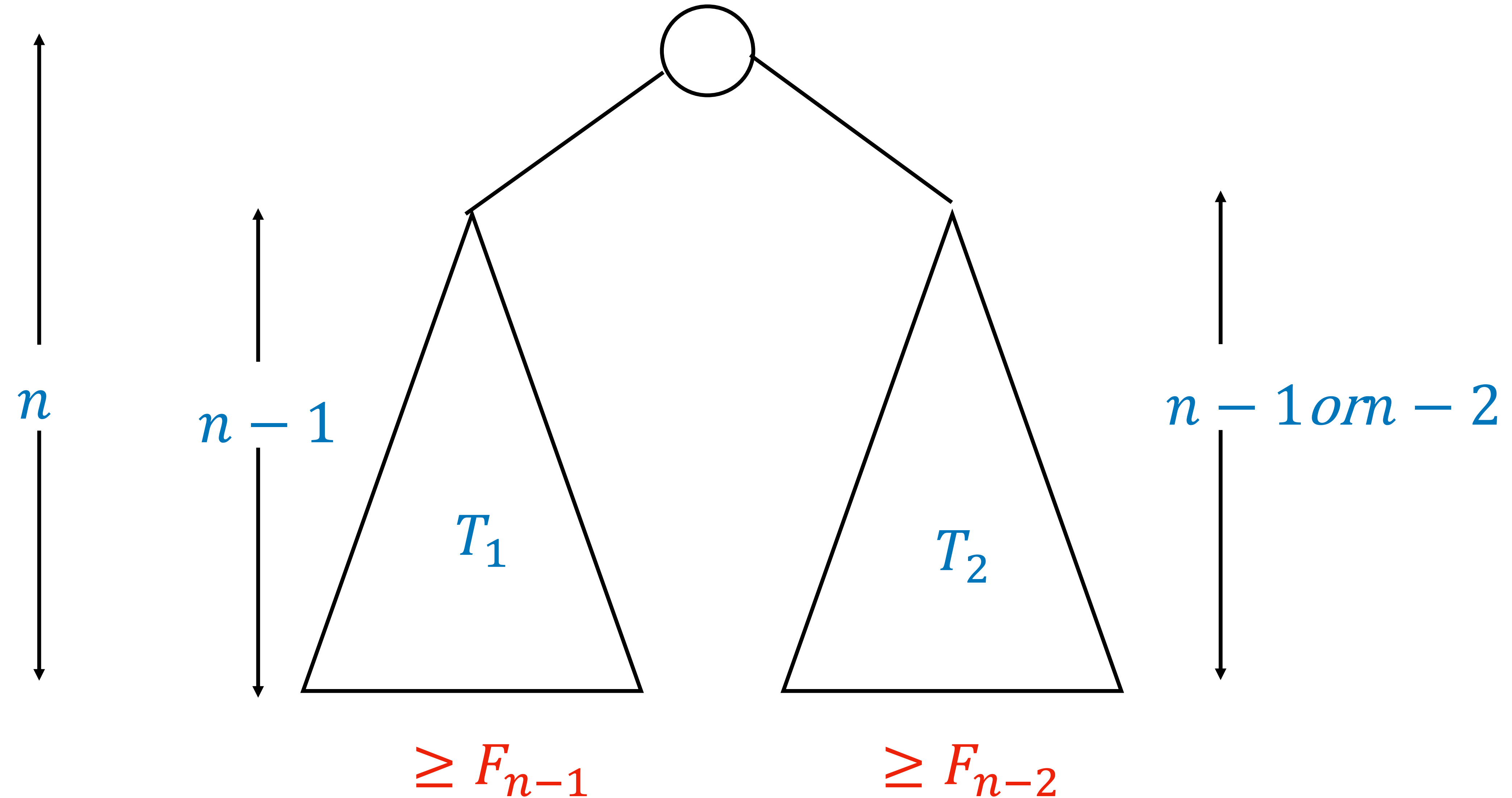
$$n \geq 2 \rightarrow F_n \geq \frac{1}{2\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n$$

balance of a node $x \in T$

$$bal(x) = h(L(x)) - h(R(x))$$

AVL-property:

$$AVL(T) \equiv \forall x \in T : bal(x) \in \{-1, 0, 1\}$$



excursion

Lemma 2. *AVL trees are balanced.*

$$|T| = n \wedge AVL(T) \quad \rightarrow \quad h(T) = O(\log n)$$

Def: *Fibonacci numbers*

$$\begin{aligned} F_0 &= 0 \\ F_1 &= 1 \\ F_n &= F_{n-1} + F_{n-2} \end{aligned}$$

$$\boxed{h(T) = n \rightarrow nl(T) = |T| + 1 \geq F_n}$$

Lemma 3. *Moivre-Binet formula*

$$F_n = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right)$$

$$n \geq 2 \rightarrow F_n \geq \frac{1}{2\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n$$

Proof. Induction on n.

$$F_0 = \frac{1}{\sqrt{5}}(1-1) = 0$$

$$F_1 = \frac{1}{\sqrt{5}}\left(\frac{2\sqrt{5}}{2}\right) = 1$$

Lemma 2. *AVL trees are balanced.*

$$|T| = n \wedge AVL(T) \quad \to \quad h(T) = O(\log n)$$

Def: *Fibonacci numbers*

$$\begin{aligned} F_0 &= 0 \\ F_1 &= 1 \\ F_n &= F_{n-1} + F_{n-2} \end{aligned}$$

$$\boxed{h(T) = n \to nl(T) = |T| + 1 \geq F_n}$$

Lemma 3. *Moivre-Binet formula*

$$F_n = \frac{1}{\sqrt{5}} \left(\left(\frac{1+\sqrt{5}}{2} \right)^n - \left(\frac{1-\sqrt{5}}{2} \right)^n \right)$$

$$n \geq 2 \to F_n \geq \frac{1}{2\sqrt{5}} \left(\frac{1+\sqrt{5}}{2} \right)^n$$

$n, n+1 \to n+2$:

$$A_n = \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2} \right)^n \quad , \quad B_n = \frac{1}{\sqrt{5}} \left(\frac{1-\sqrt{5}}{2} \right)^n$$

$$\begin{aligned} A_n + A_{n+1} &= \frac{1}{\sqrt{5}} \left(\left(\frac{1+\sqrt{5}}{2} \right)^n + \left(\frac{1+\sqrt{5}}{2} \right)^{n+1} \right) \\ &= \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2} \right)^n \left(1 + \frac{1+\sqrt{5}}{2} \right) \\ &= \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2} \right)^n \left(\frac{3+\sqrt{5}}{2} \right) \end{aligned}$$

Lemma 2. *AVL trees are balanced.*

$$|T| = n \wedge AVL(T) \quad \rightarrow \quad h(T) = O(\log n)$$

Def: *Fibonacci numbers*

$$\begin{aligned} F_0 &= 0 \\ F_1 &= 1 \\ F_n &= F_{n-1} + F_{n-2} \end{aligned}$$

$$\boxed{h(T) = n \rightarrow nl(T) = |T| + 1 \geq F_n}$$

Lemma 3. *Moivre-Binet formula*

$$F_n = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right)$$

$$n \geq 2 \rightarrow F_n \geq \frac{1}{2\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n$$

$n, n+1 \rightarrow n+2$:

$$A_n = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n \quad , \quad B_n = \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n$$

$$\begin{aligned} A_n + A_{n+1} &= \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n + \left(\frac{1+\sqrt{5}}{2}\right)^{n+1}\right) \\ &= \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n \left(1 + \frac{1+\sqrt{5}}{2}\right) \\ &= \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n \left(\frac{3+\sqrt{5}}{2}\right) \end{aligned}$$

$$\begin{aligned} \left(\frac{1+\sqrt{5}}{2}\right)^2 &= \frac{1}{4}(1+2\sqrt{5}+5) \\ &= \frac{1}{4}(6+2\sqrt{5}) \\ &= \frac{1}{2}(3+\sqrt{5}) \end{aligned}$$

excursion

Lemma 2. *AVL trees are balanced.*

$$|T| = n \wedge AVL(T) \quad \rightarrow \quad h(T) = O(\log n)$$

Def: *Fibonacci numbers*

$$\begin{aligned} F_0 &= 0 \\ F_1 &= 1 \\ F_n &= F_{n-1} + F_{n-2} \end{aligned}$$

$$\boxed{h(T) = n \rightarrow nl(T) = |T| + 1 \geq F_n}$$

Lemma 3. *Moivre-Binet formula*

$$F_n = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right)$$

$$n \geq 2 \rightarrow F_n \geq \frac{1}{2\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n$$

$n, n+1 \rightarrow n+2$:

$$A_n = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n \quad , \quad B_n = \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n$$

$$\begin{aligned} A_n + A_{n+1} &= \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n + \left(\frac{1+\sqrt{5}}{2}\right)^{n+1}\right) \\ &= \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n \left(1 + \frac{1+\sqrt{5}}{2}\right) \\ &= \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n \left(\frac{3+\sqrt{5}}{2}\right) \end{aligned}$$

$$\begin{aligned} \left(\frac{1+\sqrt{5}}{2}\right)^2 &= \frac{1}{4}(1 + 2\sqrt{5} + 5) \\ &= \frac{1}{4}(6 + 2\sqrt{5}) \\ &= \frac{1}{2}(3 + \sqrt{5}) \end{aligned}$$

$$\begin{aligned} A_n + A_{n+1} &= \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n \left(\frac{1+\sqrt{5}}{2}\right)^2 \\ &= A_{n+2} \\ B_n + B_{n+1} &= B_{n+2} \quad \text{similarly} \end{aligned}$$

rebalancing trees after insertion or deletion of node y

- locate place to insert or delete node y for BST's
- rebalance bottom up on path from y to root
- use rotations and double rotations for this

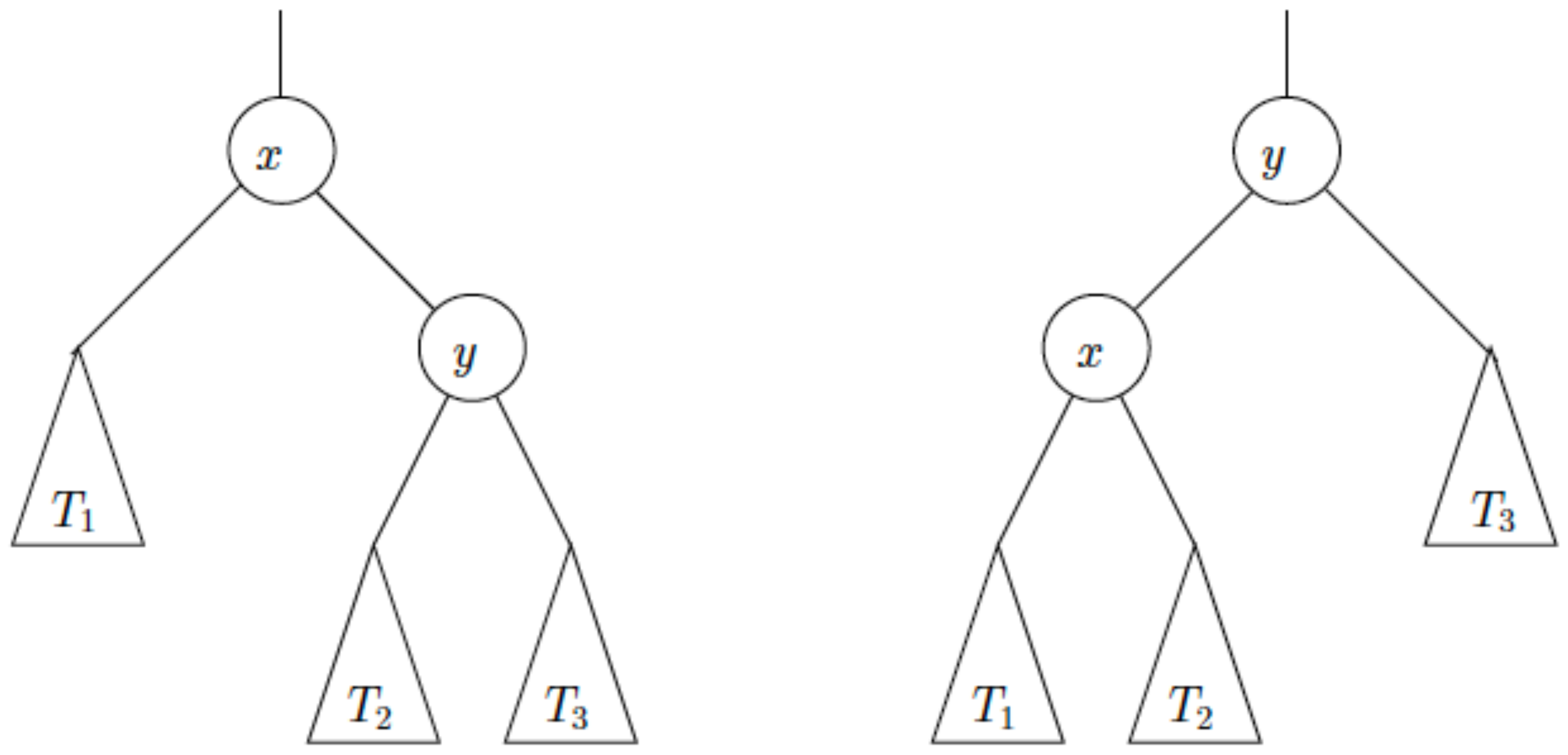


Figure 7.1: A left rotation around x transforms the tree on the left hand side into the tree on the right hand side. A right rotation around y transforms the tree on the right hand side into the tree on the left hand side.

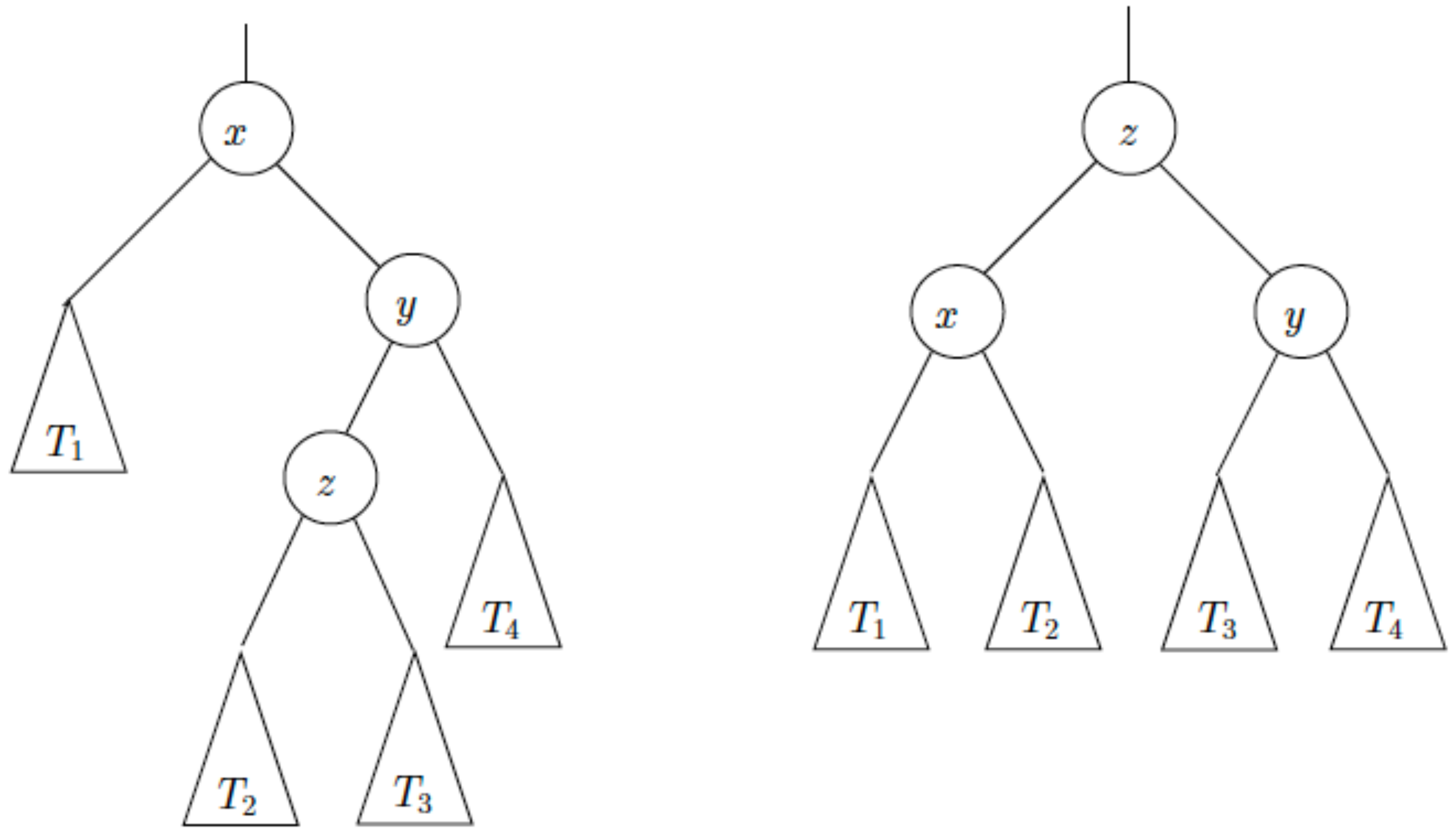


Figure 7.2: A left double rotation. You can think of a double left rotation as a right rotation around y followed by a left rotation around x. This explains the name double rotation. A double rotation also preserve the binary search tree property.

rebalancing trees after insertion of node y

- locate place to insert node y for BST’s
- rebalance bottom up on path from y to root
- use rotations and double rotations for this

Algorithm 31 AVL-insert-repair

Input: AVL tree T, a node y inserted into T

Output: afterwards, the AVL property is restored

$x :=$ Parent(y)
while $x \neq$ NULL **do** $\quad$ $y.p \neq null, y\, is\ not\ root$
  if $y =$ Left(x) **then**
    Bal$(x) :=$ Bal$(x) + 1$
  else
    Bal$(x) :=$ Bal$(x) - 1$
  if Bal$(x) = 0$ **then**
    return
  if Bal$(x) = 2$ or Bal$(x) = -2$ **then**
    Restore the AVL property using a rotation or double rotation (see Figure 7.1 and 7.2)
    return
  $y := x$; $x :=$ Parent(y) $\quad$ bottom up

rebalancing trees after insertion of node y

- locate place to insert node y for BST's
- rebalance bottom up on path from y to root
- use rotations and double rotations for this

Algorithm 31 AVL-insert-repair

Input: AVL tree T, a node y inserted into T

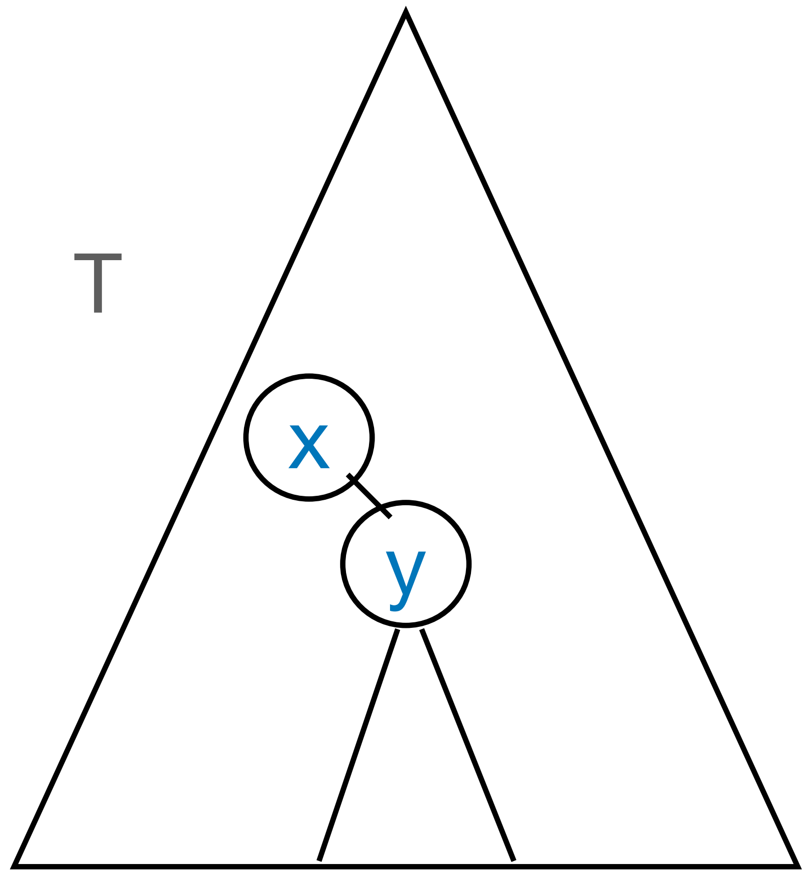
Output: afterwards, the AVL property is restored

```
x := Parent(y)
while x ≠ NULL do                         y.p ≠ null, y is not root
    if y = Left(x) then
        Bal(x) := Bal(x) + 1
    else
        Bal(x) := Bal(x) − 1
    if Bal(x) = 0 then
        return
    if Bal(x) = 2 or Bal(x) = −2 then
        Restore the AVL property using a rotation or double rotation (see
        Figure 7.1 and 7.2)                rotation only once
        return
    y := x; x := Parent(y)                 bottom up
```

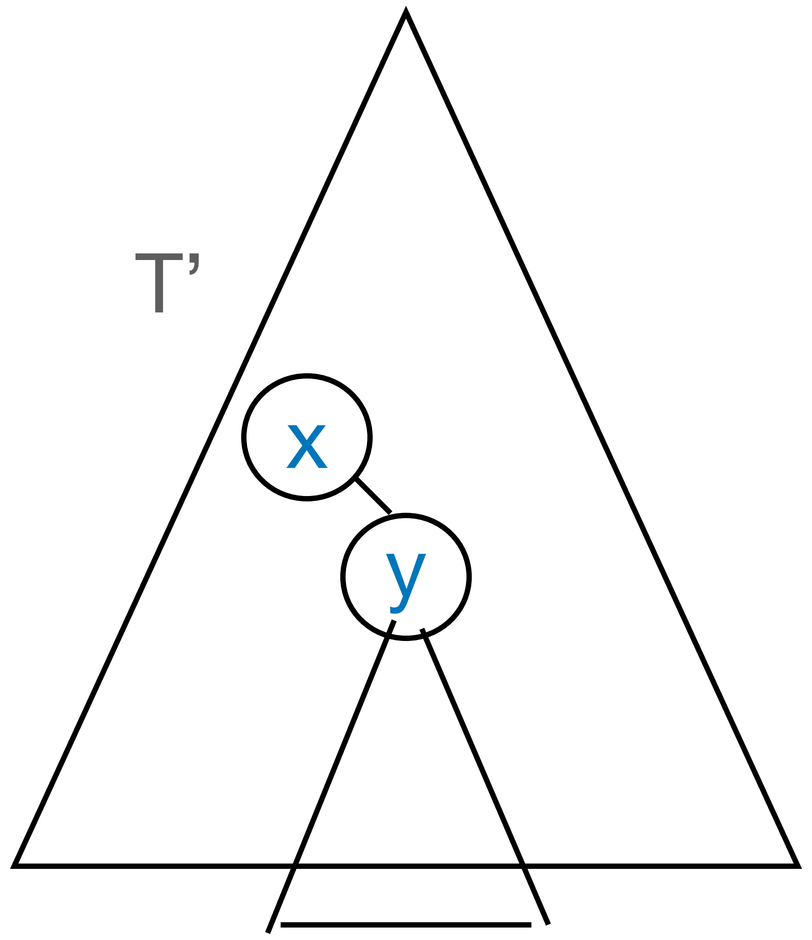
for correctness of 1 pass of loop
we must argue about 3 trees

- T with l ,r ,p, h, bal in original tree
- T' with l', r', p', h', bal' after insertion of y
- T'' after rotation

T



T'



notation:

$$\tilde{y} = \begin{cases} l(p'(y)) & isl(y) \\ r(p'(y)) & isr(y) \end{cases}$$

the node in the original tree whose place was taken by y. We have $\tilde{y} \neq y$ after rotations.

rebalancing trees after insertion of node y

- locate place to insert node y for BST's
- rebalance bottom up on path from y to root
- use rotations and double rotations for this

for correctness of 1 pass of loop
we must argue about 3 trees

- T with l ,r ,p, h, bal in original tree
- T' with l', r', p', h', bal' after insertion
- T'' after rotation

Algorithm 31 AVL-insert-repair

Input: AVL tree T, a node y inserted into T

Output: afterwards, the AVL property is restored

$x :=$ Parent(y)

while $x \neq$ NULL **do** $\quad y.p \neq null, y\, is\, not\, root$

  if $y =$ Left(x) **then**

    Bal$(x) :=$ Bal$(x) + 1$

  else

    Bal$(x) :=$ Bal$(x) - 1$

  if Bal$(x) = 0$ **then**

    return h(x) unchanged

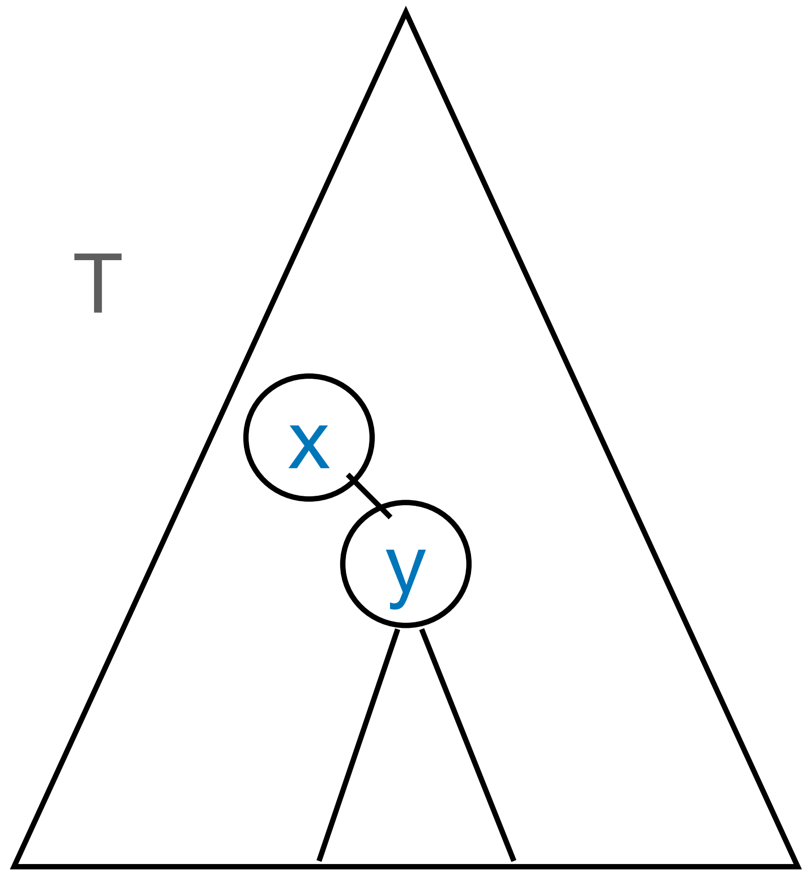
  if Bal$(x) = 2$ or Bal$(x) = -2$ **then**

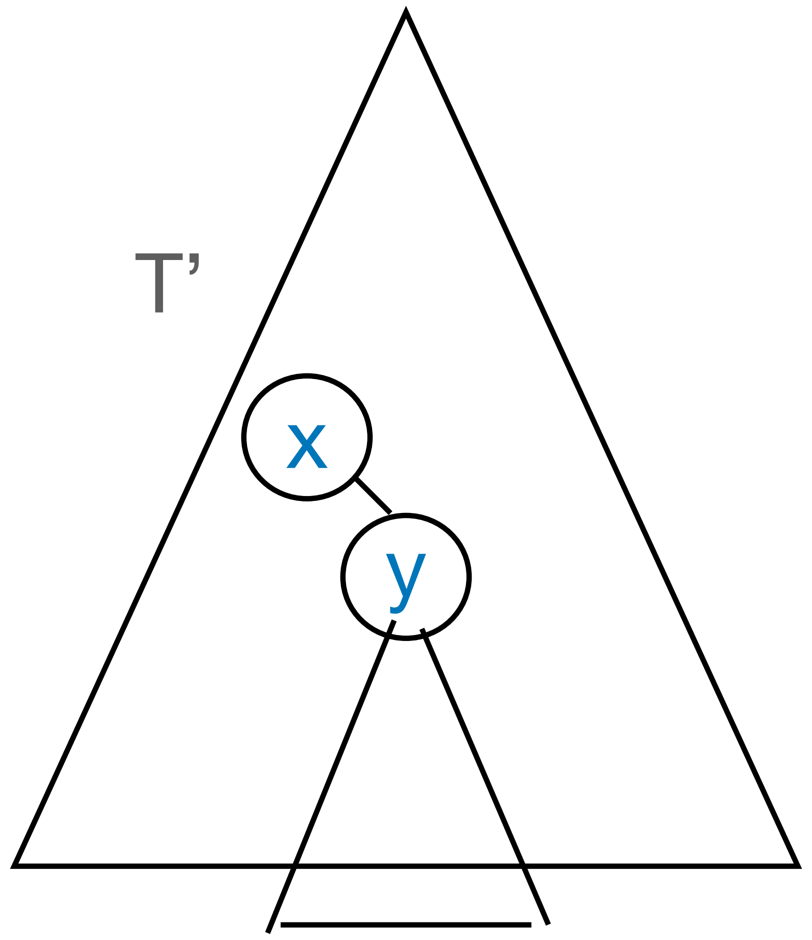
    Restore the AVL property using a rotation or double rotation (see Figure 7.1 and 7.2)

    return rotation only once

  $y := x$; $x :=$ Parent(y) bottom up

$$\boxed{bal'(x) = \begin{cases} bal(x)+1 & y = l(x) \\ bal(x)-1 & y = r(x) \end{cases}}$$





rebalancing trees after insertion of node y

- locate place to insert node y for BST's
- rebalance bottom up on path from y to root
- use rotations and double rotations for this

Algorithm 31 AVL-insert-repair

Input: AVL tree T, a node y inserted into T

Output: afterwards, the AVL property is restored

```
x := Parent(y)
while x ≠ NULL do                         y.p ≠ null, y is not root
    if y = Left(x) then
        Bal(x) := Bal(x) + 1
    else
        Bal(x) := Bal(x) − 1
    if Bal(x) = 0 then
        return                            h(x) unchanged
    if Bal(x) = 2 or Bal(x) = −2 then
        Restore the AVL property using a rotation or double rotation (see
        Figure 7.1 and 7.2)
        return
    y := x; x := Parent(y)                bottom up
```

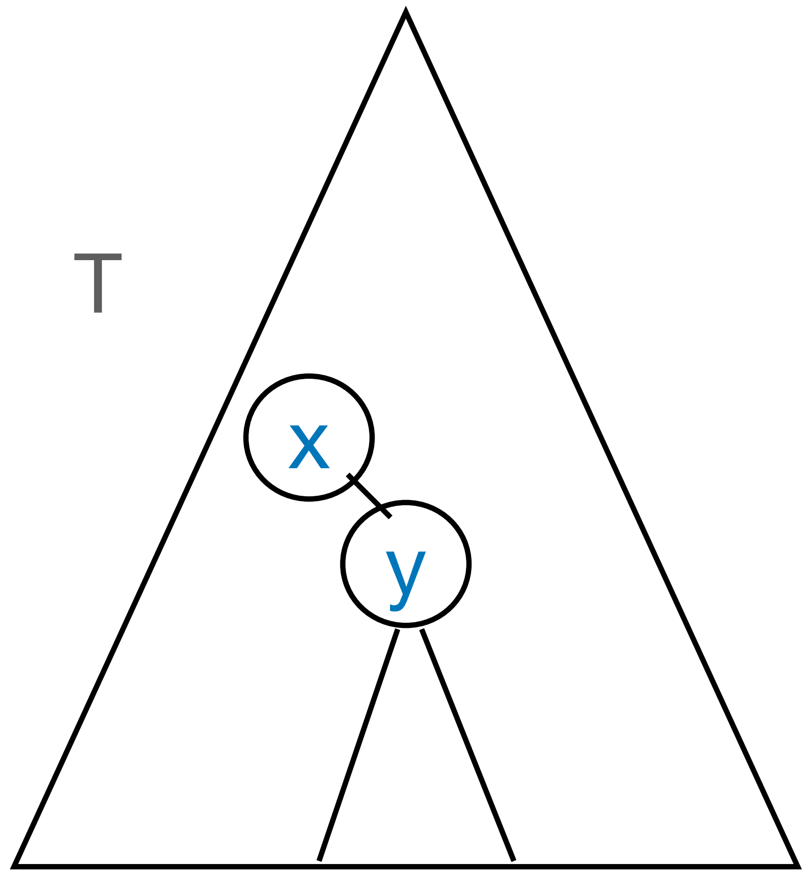
$$bal'(x) = \begin{cases} bal(x)+1 & y = l(x) \\ bal(x)-1 & y = r(x) \end{cases}$$

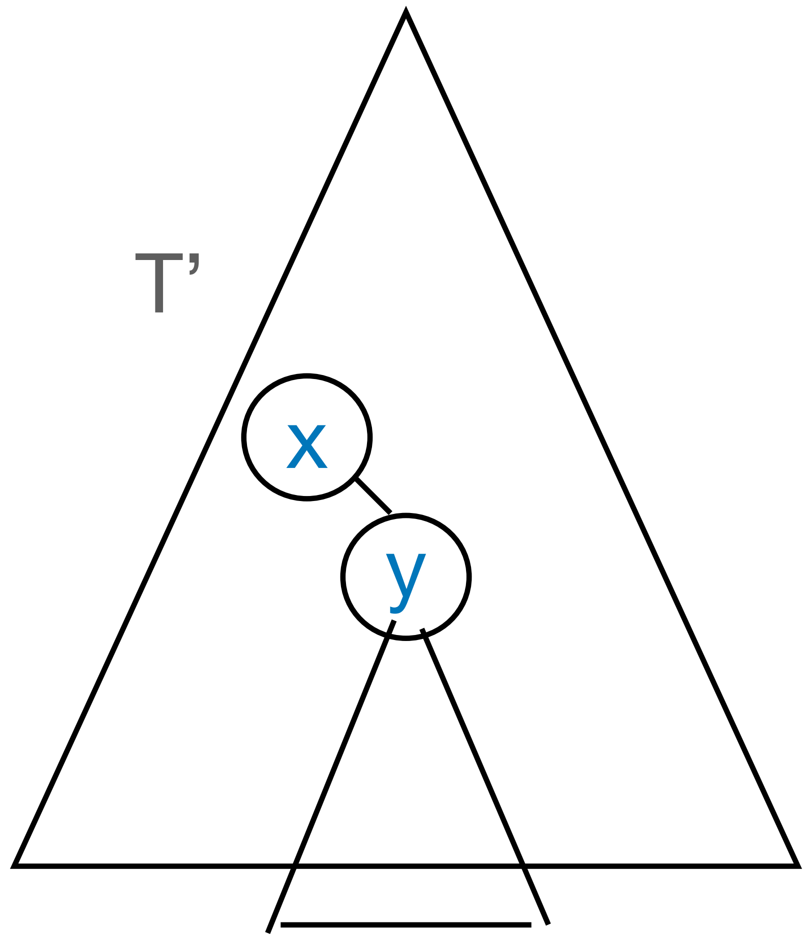
invariant when loop body is entered

$$h'(y) = h(y) + 1$$
$$bal(y) \in \{-1,1\}$$

for correctness of 1 pass of loop
we must argue about 3 trees

- T with l ,r ,p, h, bal in original tree
- T' with l', r', p', h', bal' after insertion
- T'' after rotation



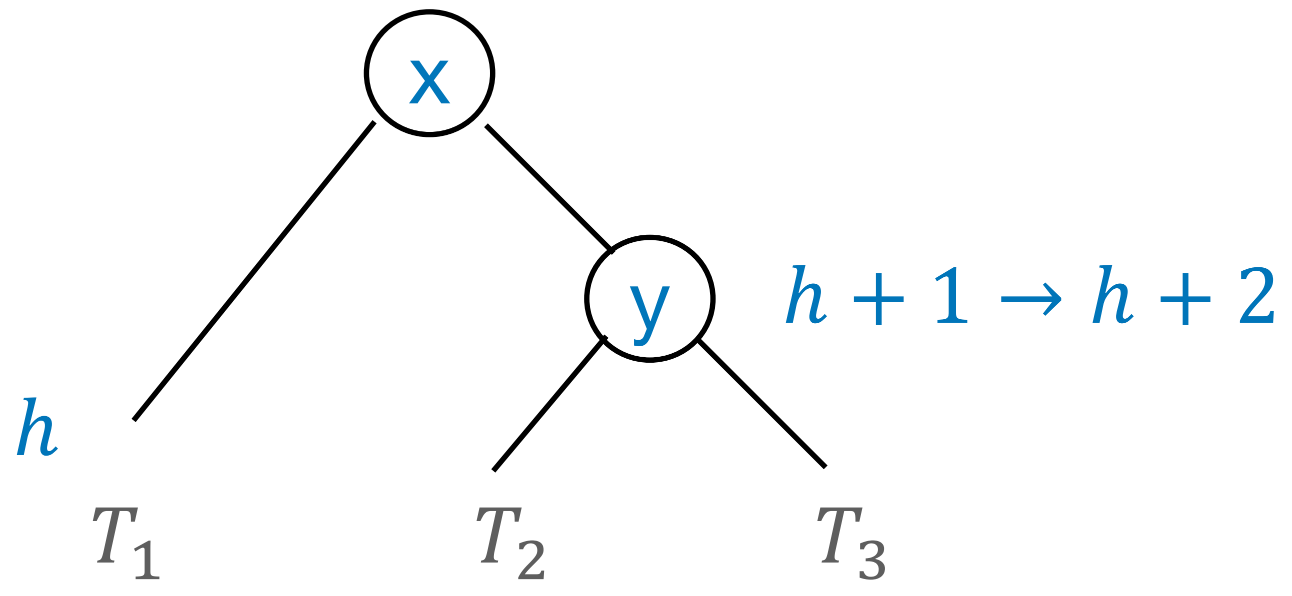


correctness

- here only $bal'(x) = -2$
  - case $bal'(y) = -1$

insertion in T_3

to show:
rotations restore AVL property for $bal'(x) \in \{-2,2\}$

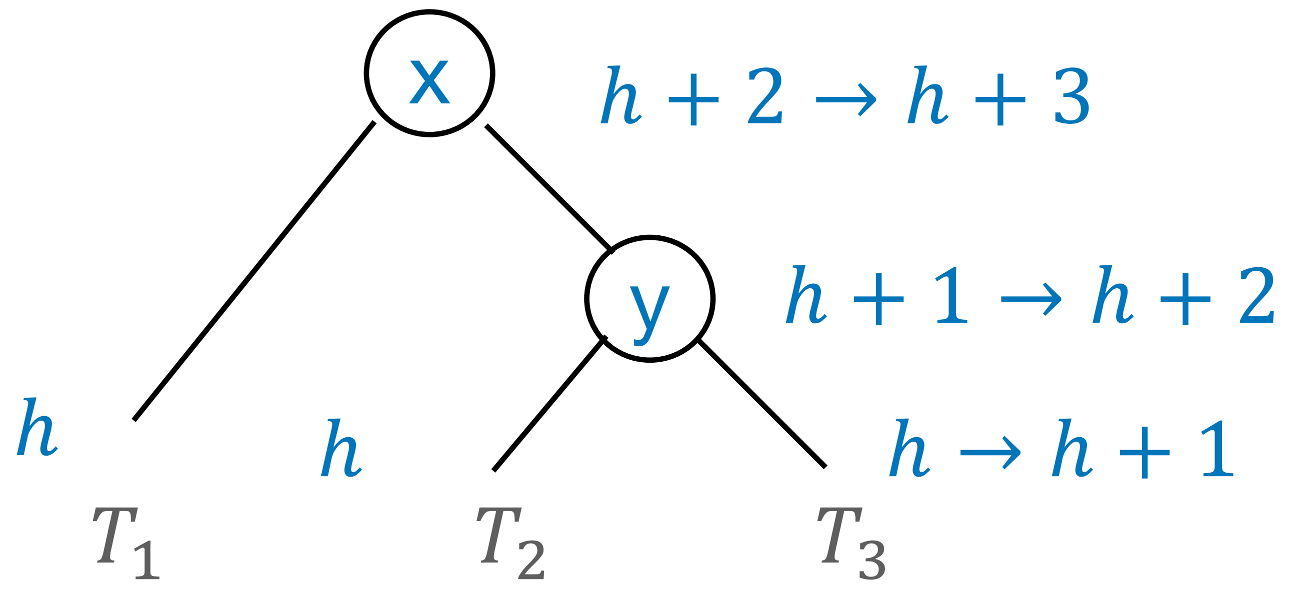


correctness

- here only $bal'(x) = -2$
  - case $bal'(y) = -1$

insertion in T_3

to show:
rotations restore AVL property for $bal'(x) \in \{-2,2\}$



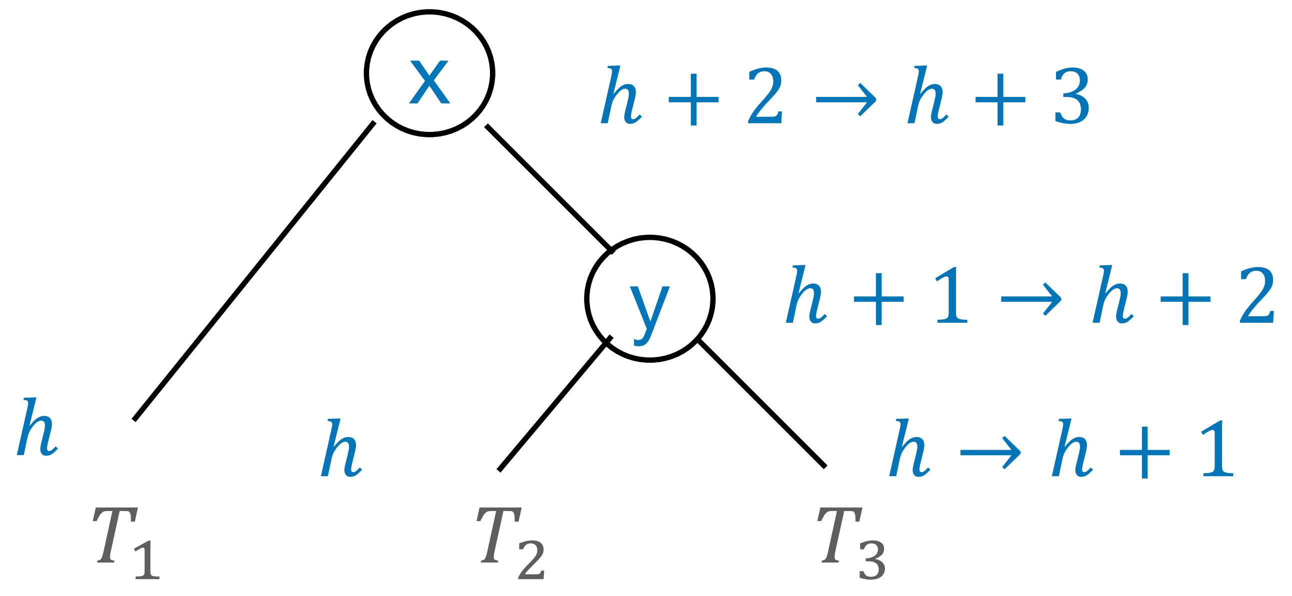
correctness

- here only $bal'(x) = -2$
  - case $bal'(y) = -1$

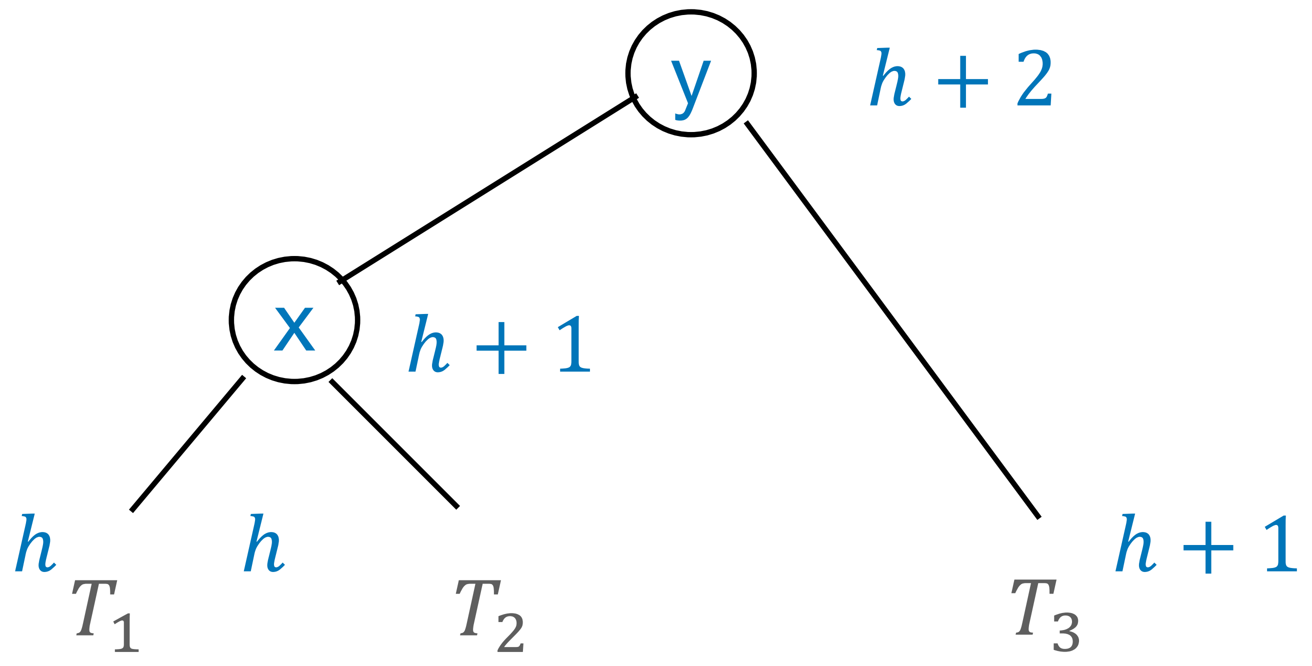
    insertion in T_3

to show:

rotations restore AVL property for $bal'(x) \in \{-2,2\}$



rotate left

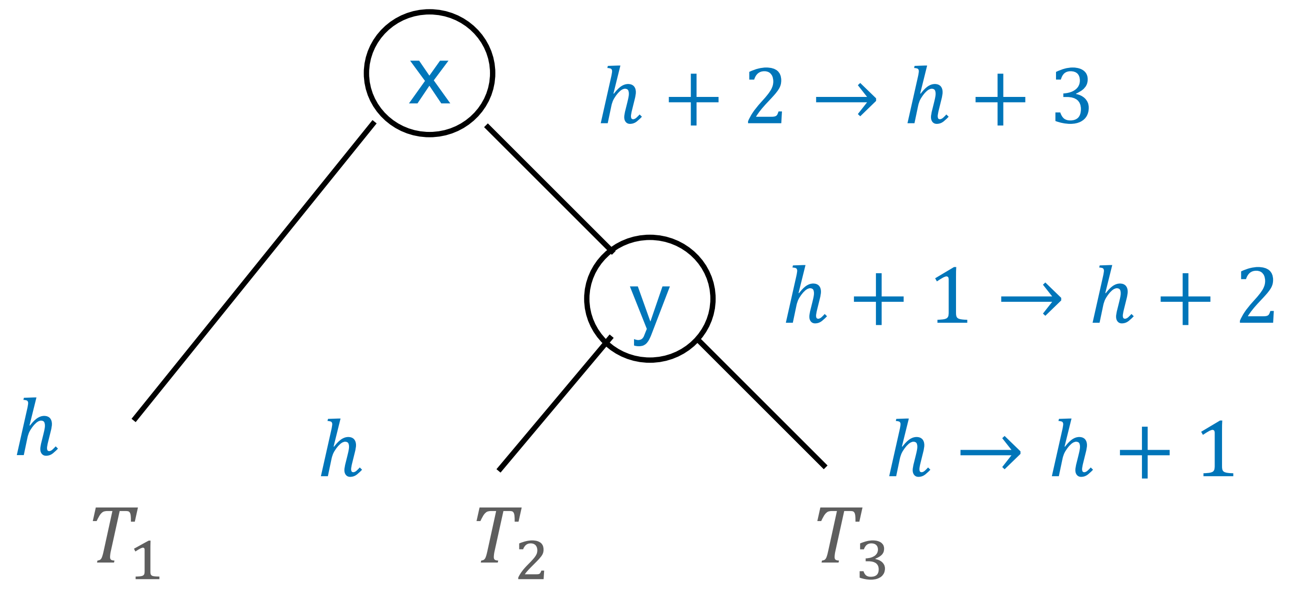


correctness

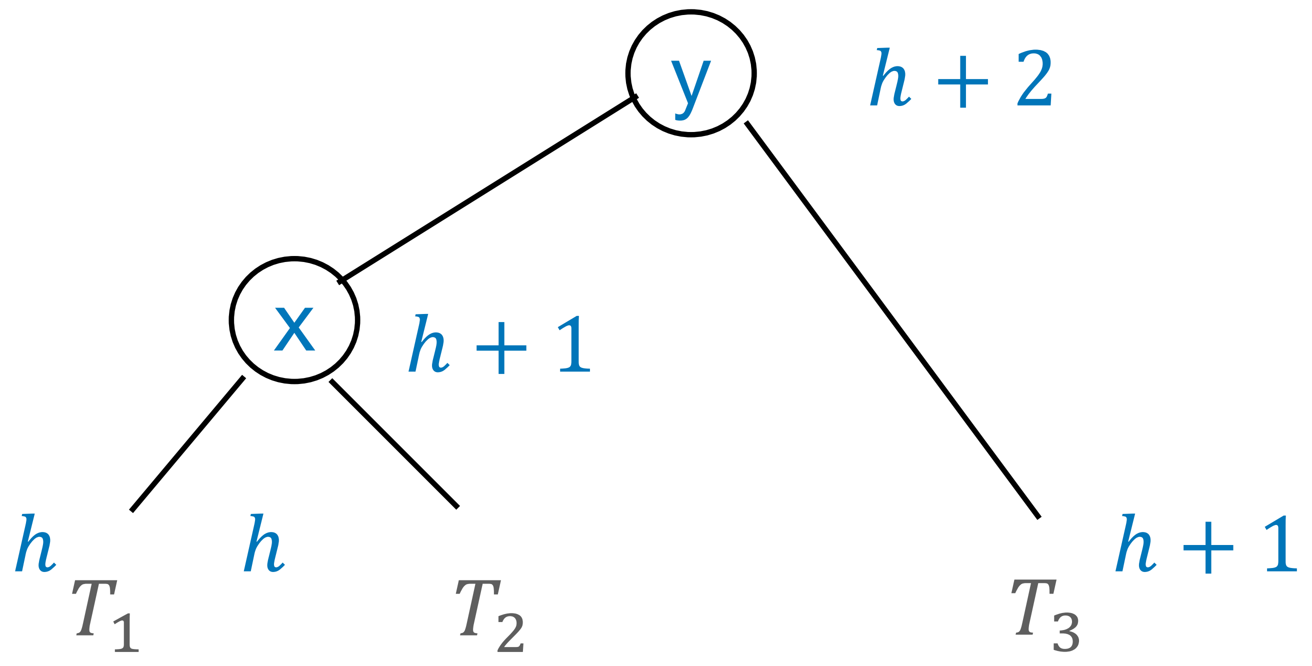
- here only $bal'(x) = -2$
  - case $bal'(y) = -1$

    insertion in T_3

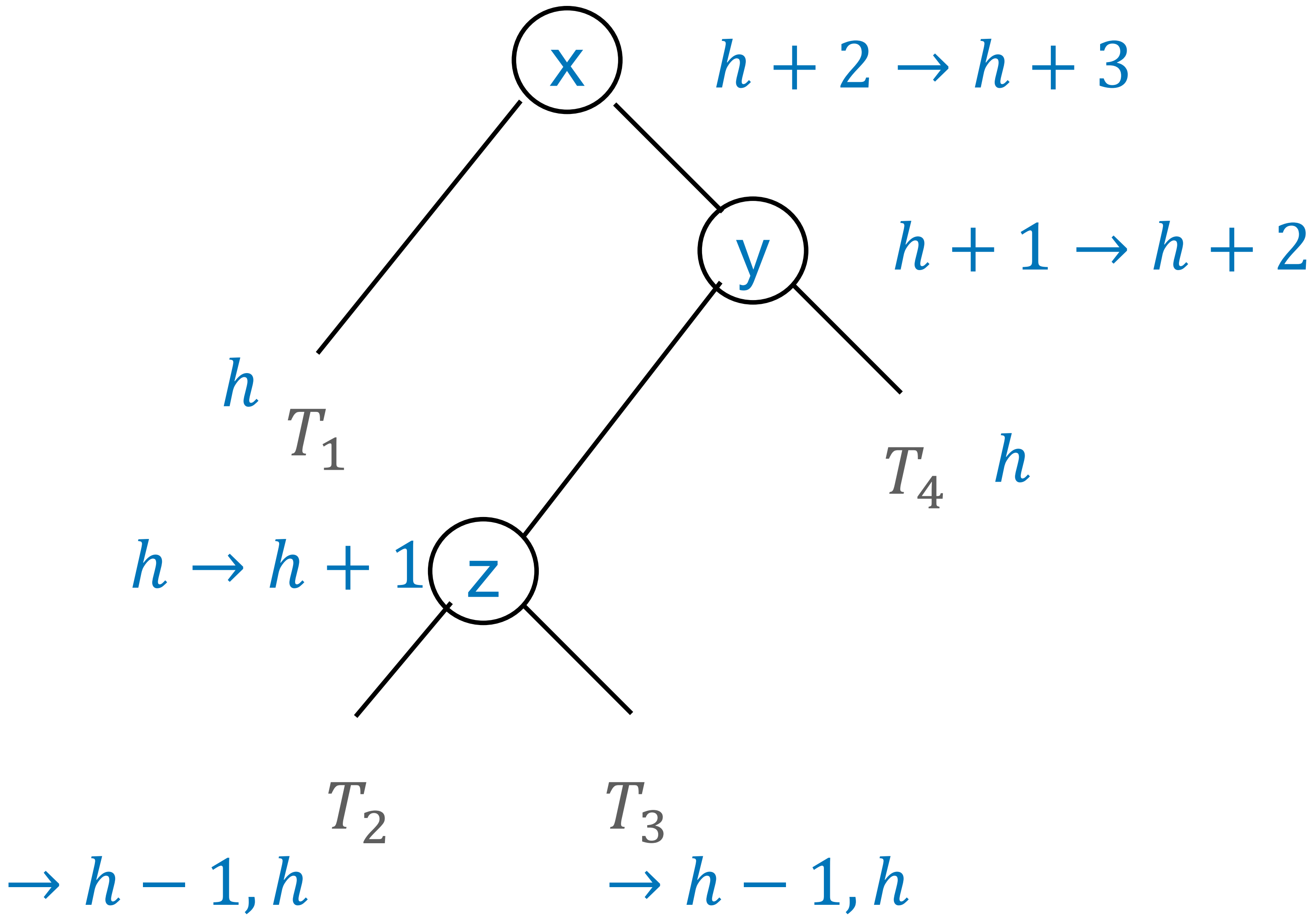
to show:
rotations restore AVL property for $bal'(x) \in \{-2,2\}$



rotate left



  - case $bal'(y) = 1$ insertion in T_2 or T_3

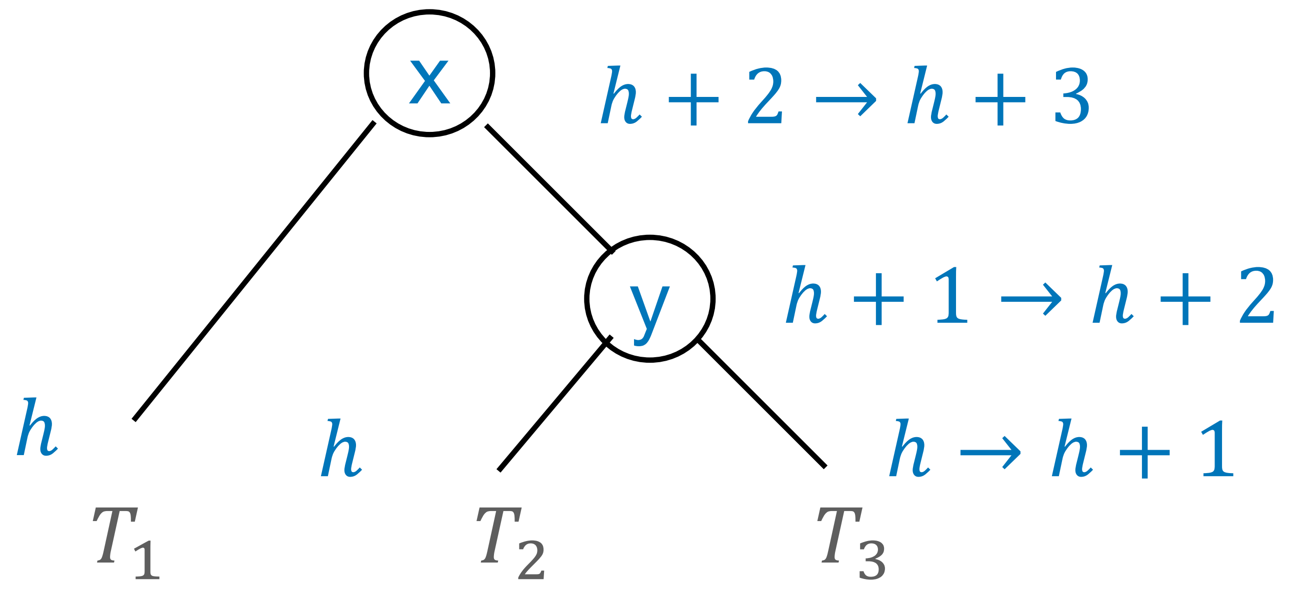


correctness

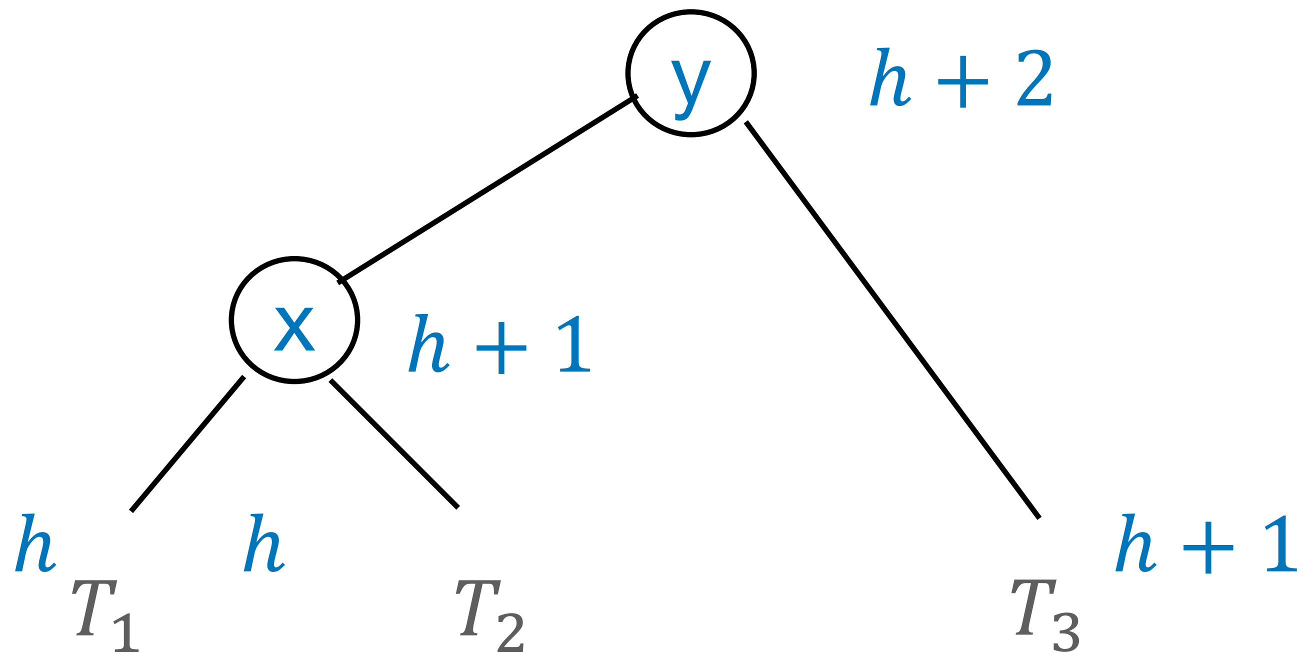
to show:
rotations restore AVL property for $bal'(x) \in \{-2,2\}$

- here only $bal'(x) = -2$
  - case $bal'(y) = -1$

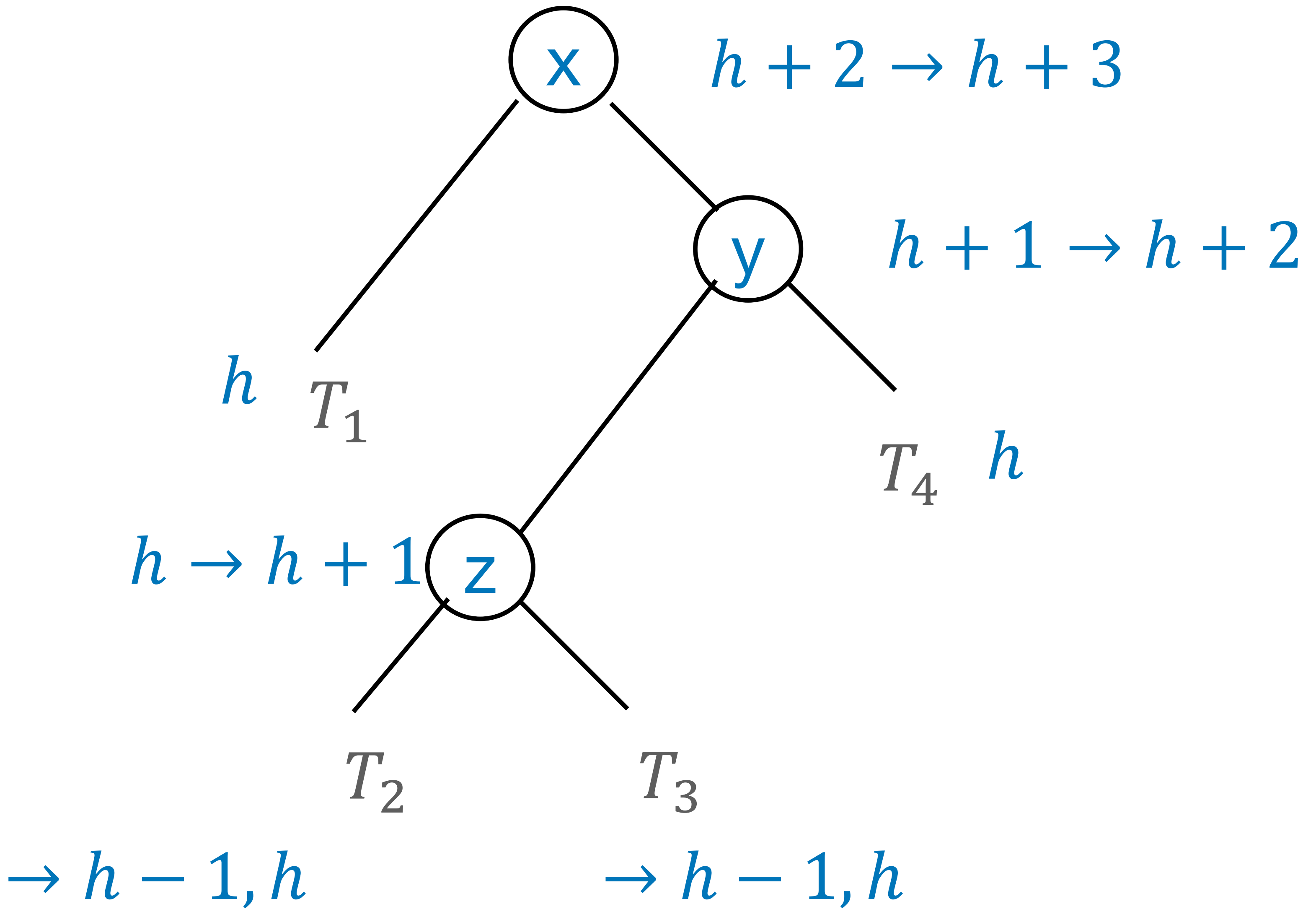
    insertion in T_3



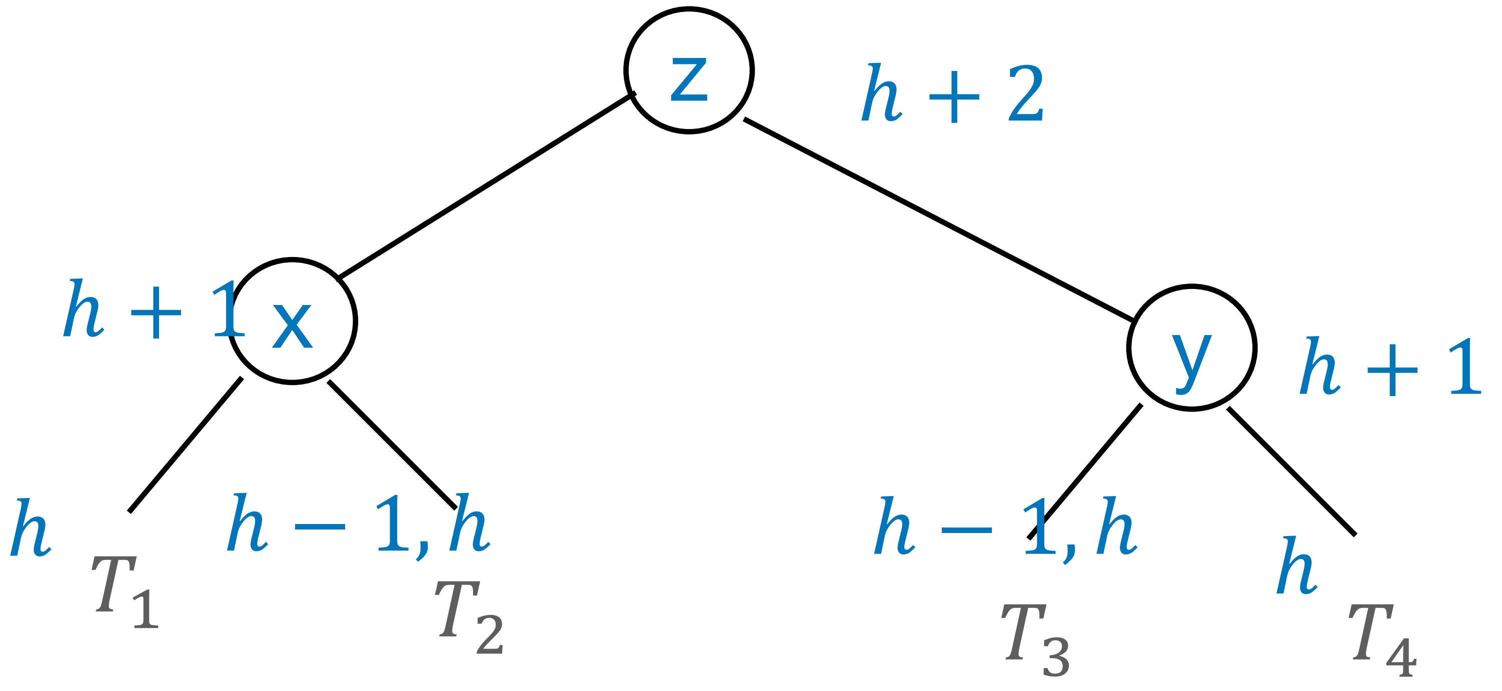
rotate left



  - case $bal'(y) = 1$ insertion in T_2 or T_3



double rotate



rebalancing trees after deletion of node y

- locate place to delete node y for BST's
- rebalance bottom up on path from y to root
- use rotations and double rotations for this

Algorithm 32 AVL-delete-repair

Input: AVL tree T, a node v deleted from T

Output: afterwards, the AVL property is restored

```
x := Parent(v)
while x ≠ NULL do
    if v = Left(x) then
        Bal(x) := Bal(x) − 1
    else
        Bal(x) := Bal(x) + 1
    if Bal(x) = 1 or Bal(x) = −1 then
        return
    if Bal(x) = 2 or Bal(x) = −2 then
        Restore the AVL property using a rotation or double rotation
    v := x; x := Parent(v)
```

$y.p \neq null, y \text{ is not root}$

$$\boxed{bal'(x) = \begin{cases} bal(x) - 1 & y = l(x) \\ bal(x) + 1 & y = r(x) \end{cases}}$$

balance was 0, height of 1 subtree deceased, h(x) unchanged

rotation at multiple levels possible

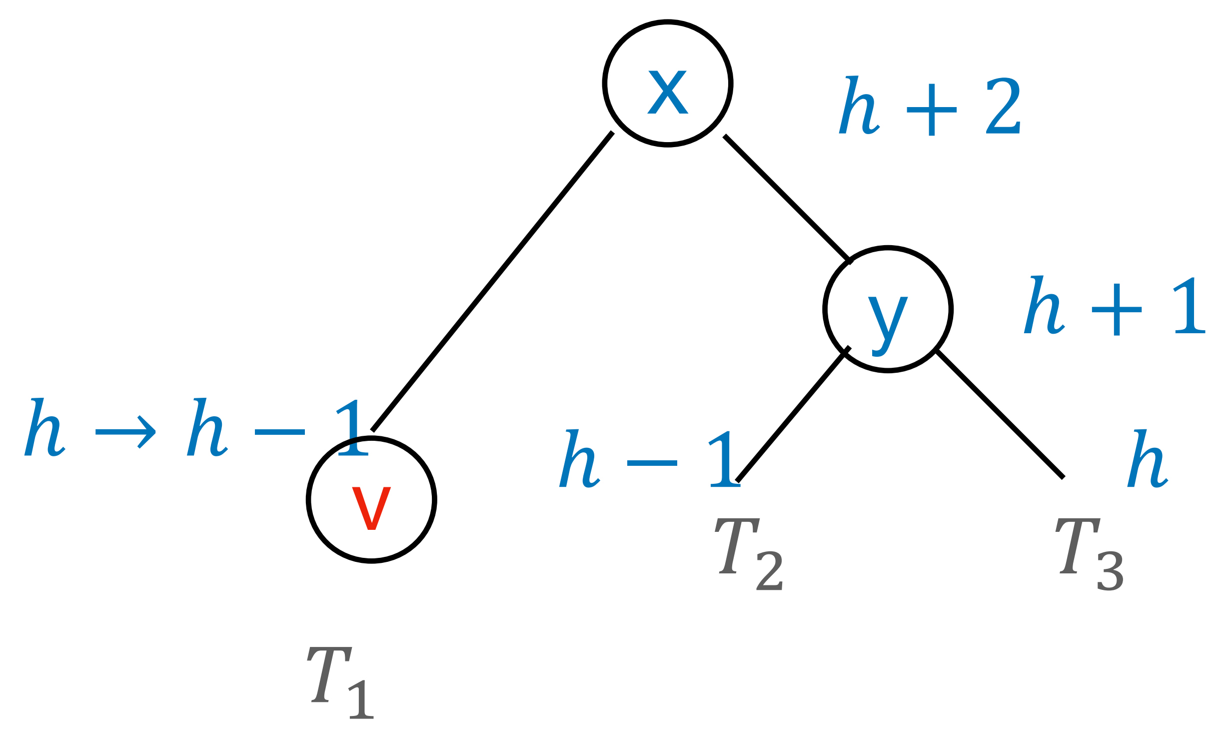
bottom up

correctness

- here only $bal'(x) = -2$
  - case $bal'(y) = -1$

deletion in T_1

to show:
rotations restore AVL property for $bal'(x) \in \{-2,2\}$

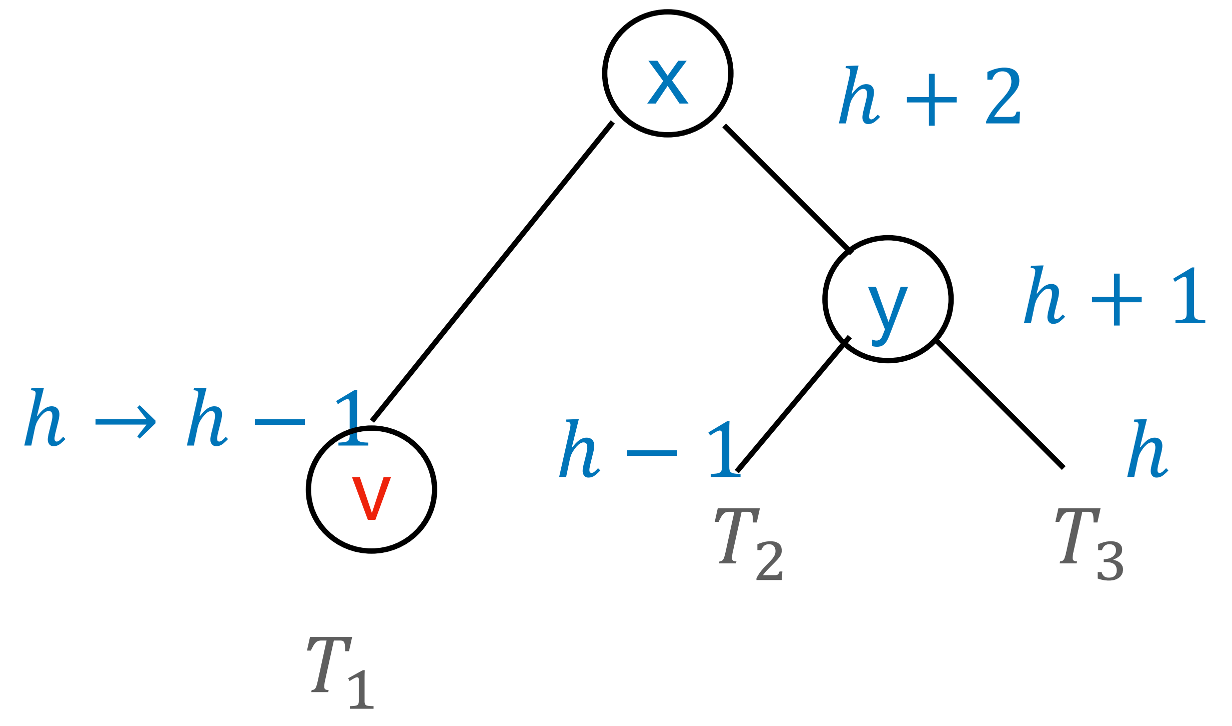


correctness

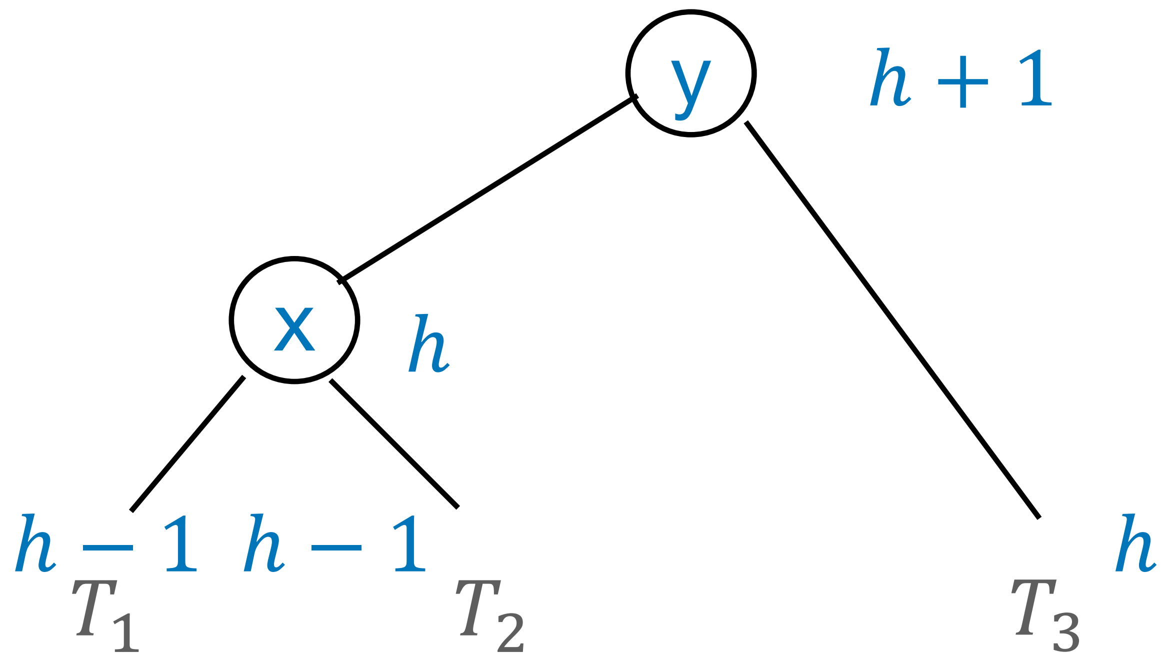
- here only $bal'(x) = -2$
    - case $bal'(y) = -1$

deletion in T_1

to show:
rotations restore AVL property for $bal'(x) \in \{-2,2\}$



rotate left



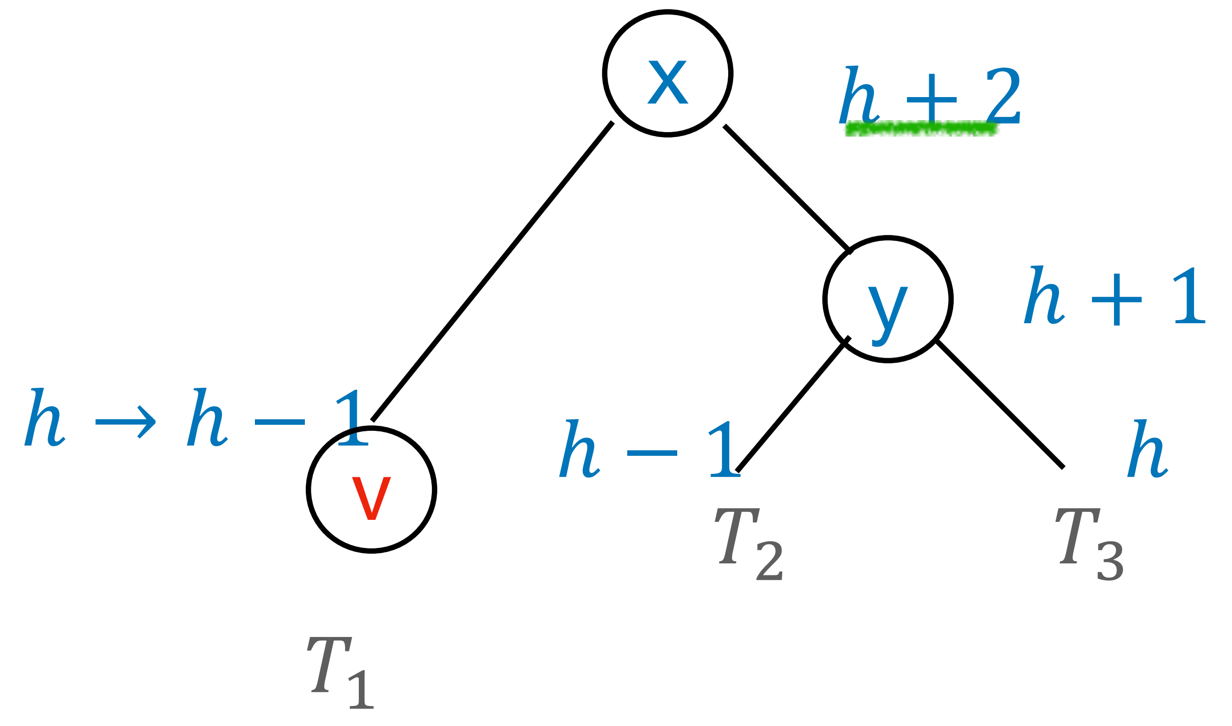
correctness

- here only $bal'(x) = -2$
  - case $bal'(y) = -1$

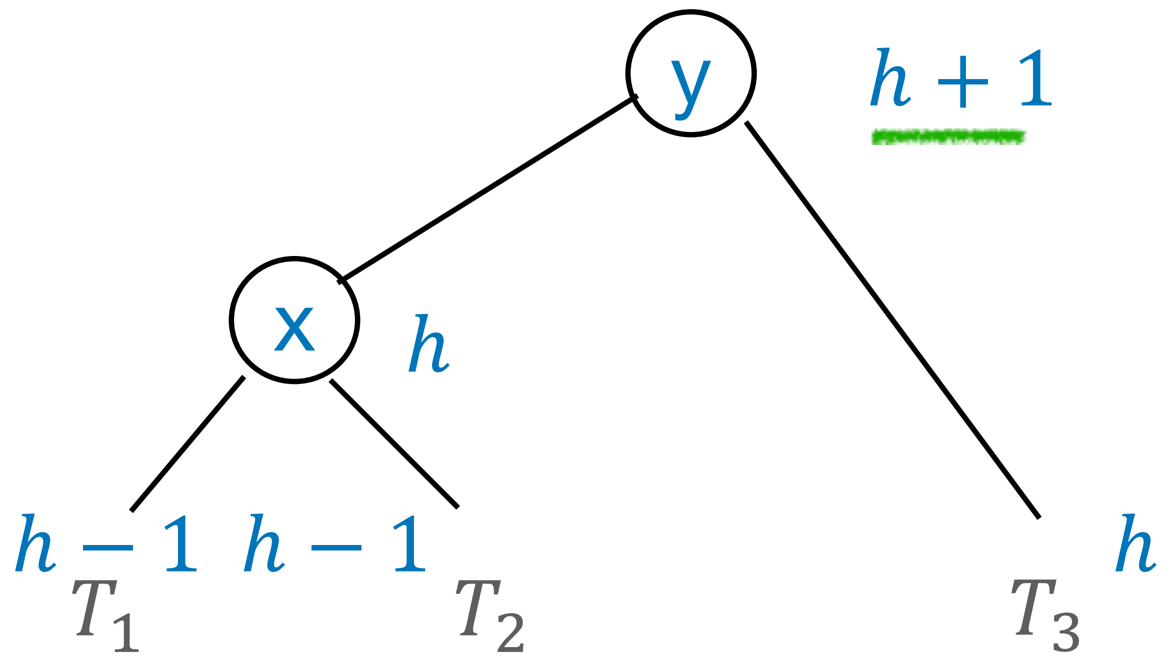
deletion in T_1

to show:
rotations restore AVL property for $bal'(x) \in \{-2,2\}$

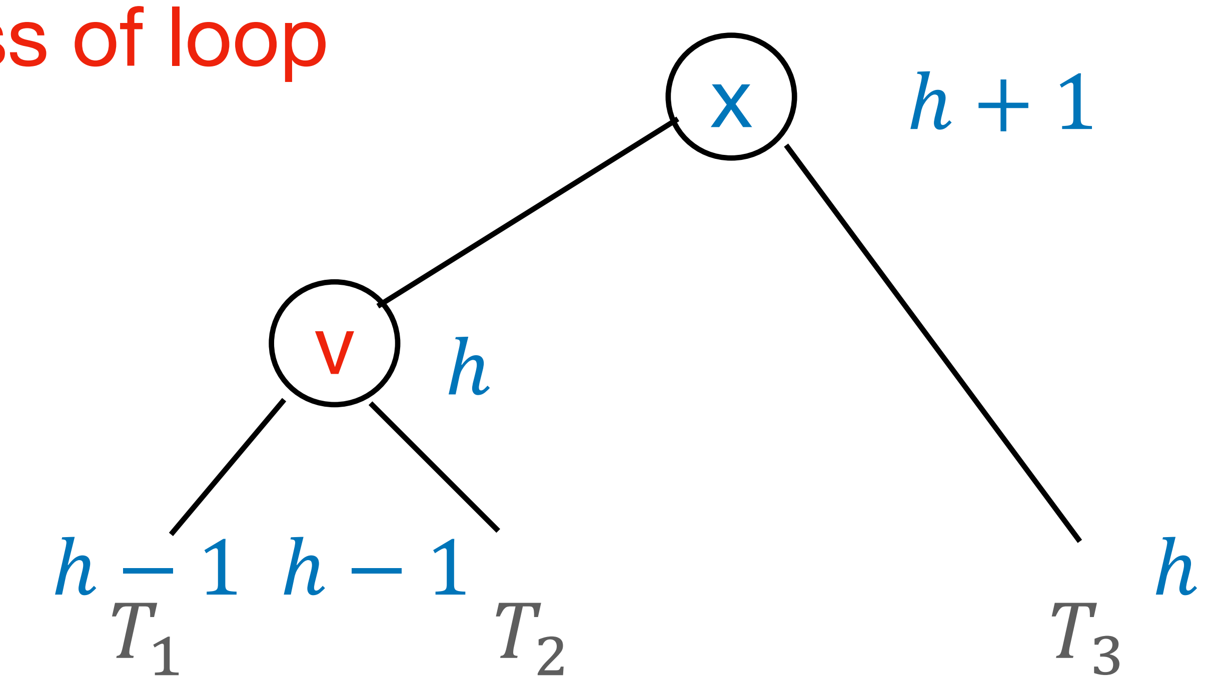
height decreased



rotate left



next pass of loop

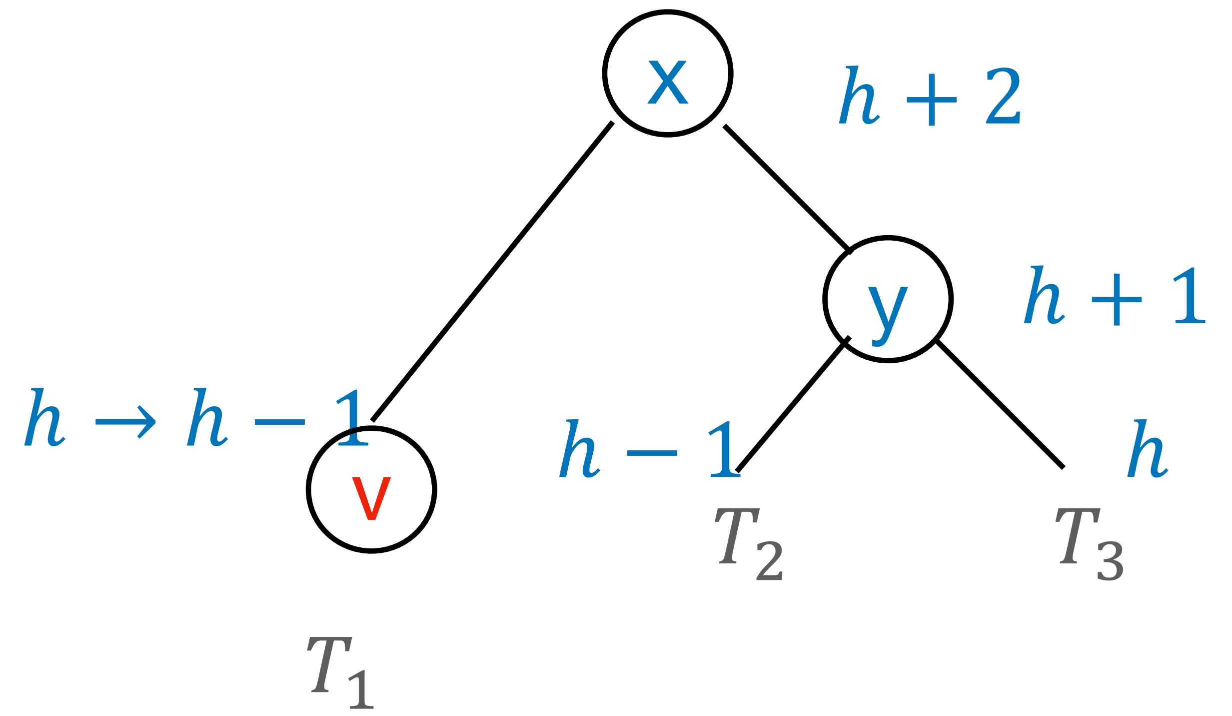


correctness

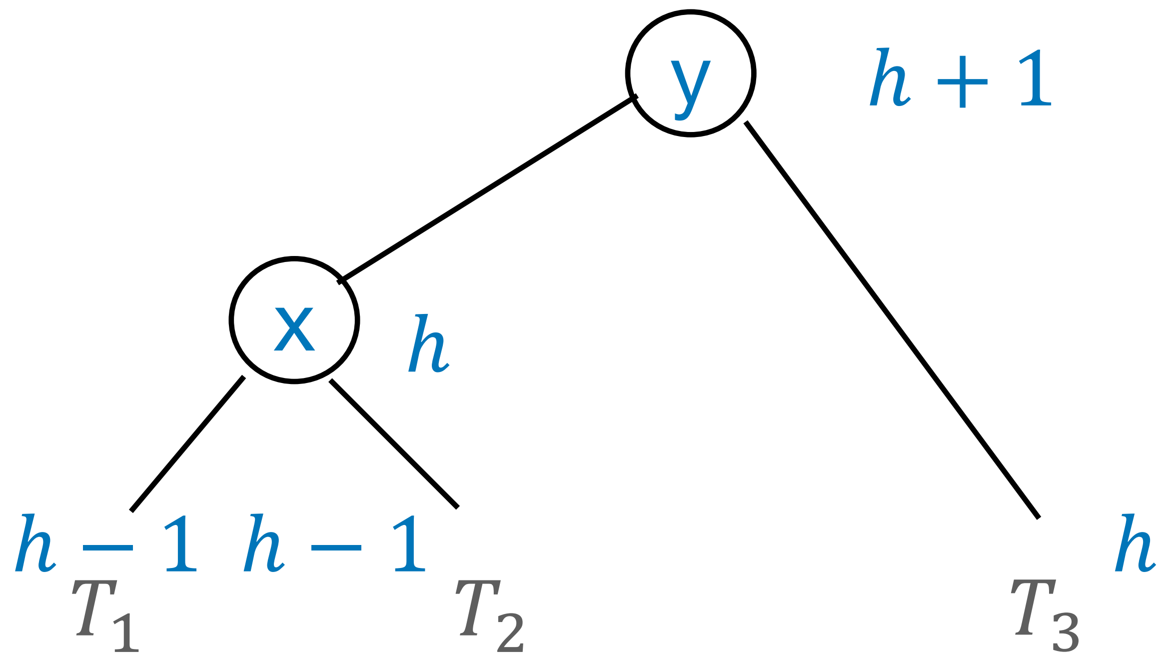
- here only $bal'(x) = -2$
  - case $bal'(y) = -1$

    deletion in T_1

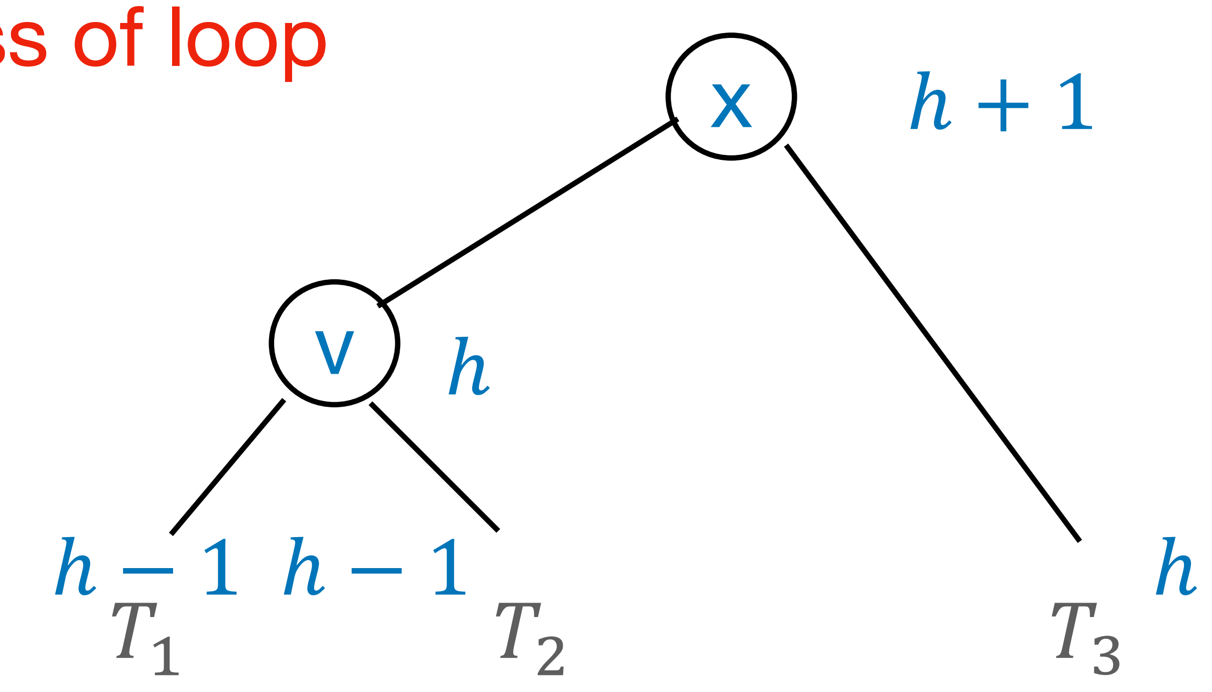
to show:
rotations restore AVL property for $bal'(x) \in \{-2,2\}$



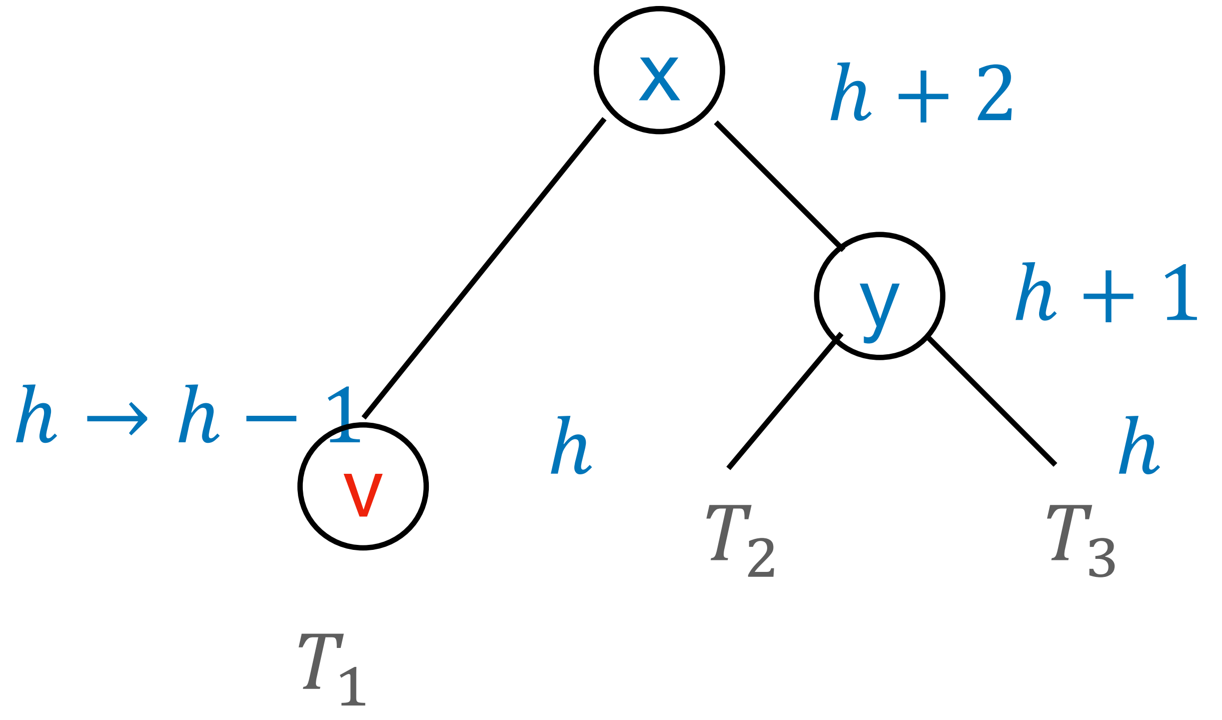
rotate left



next pass of loop



- case $bal'(y) = 0$



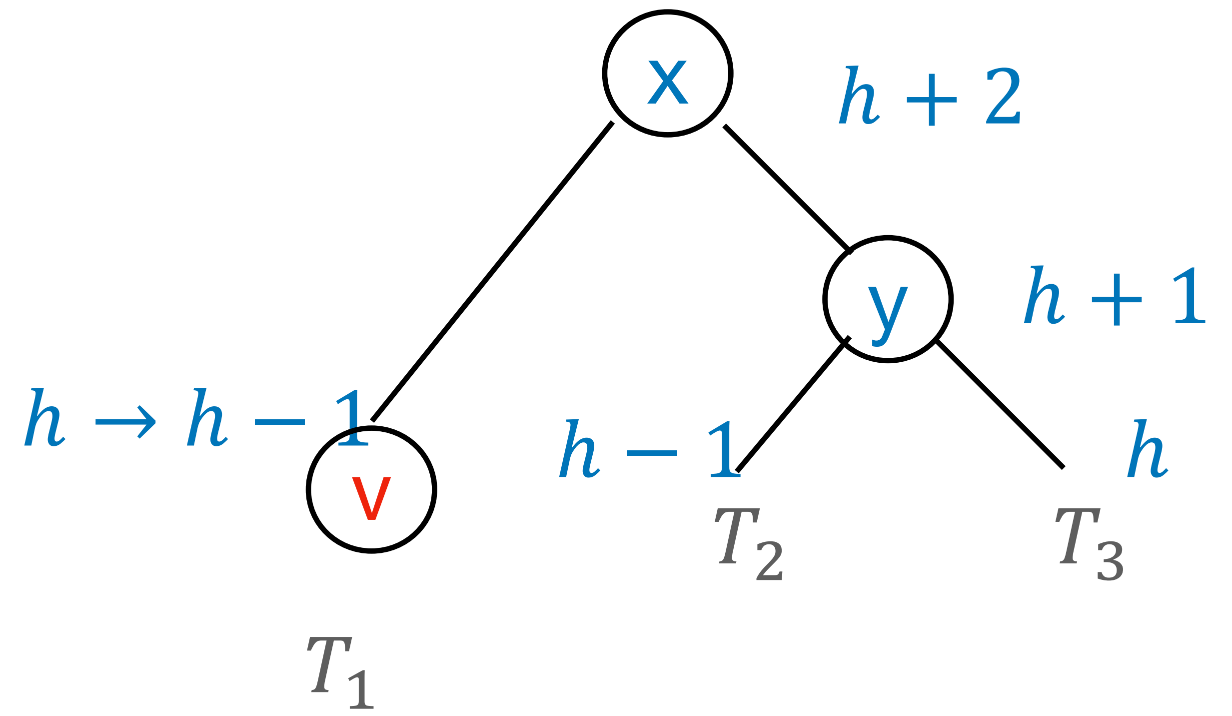
correctness

- here only $bal'(x) = -2$
  - case $bal'(y) = -1$

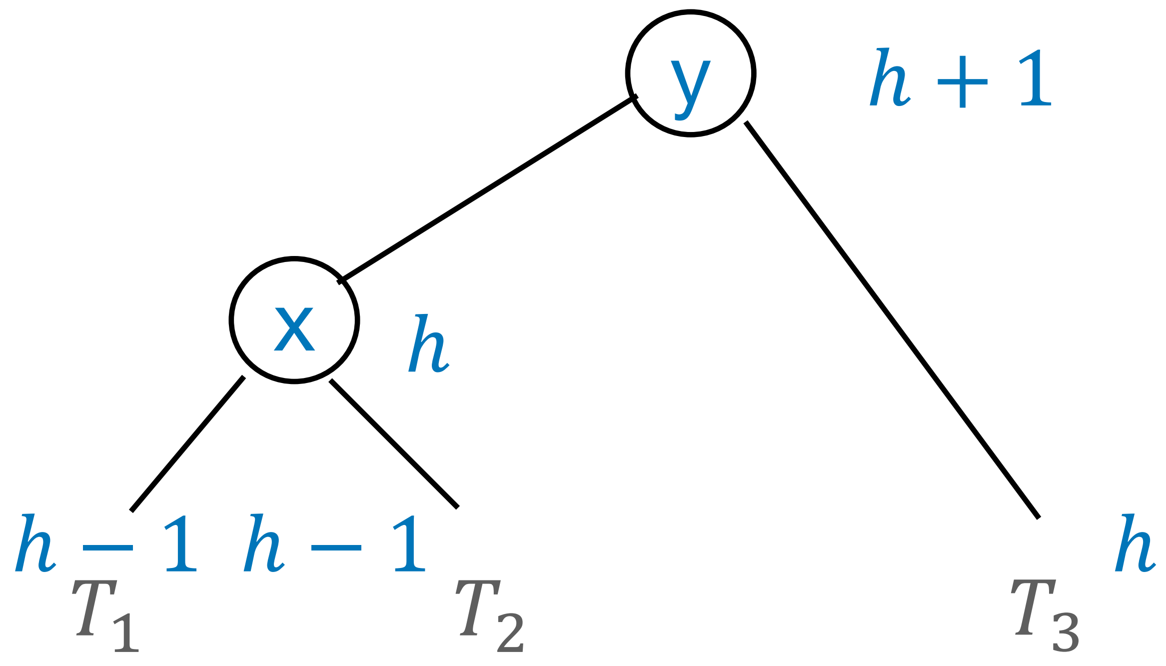
to show:

rotations restore AVL property for $bal'(x) \in \{-2,2\}$

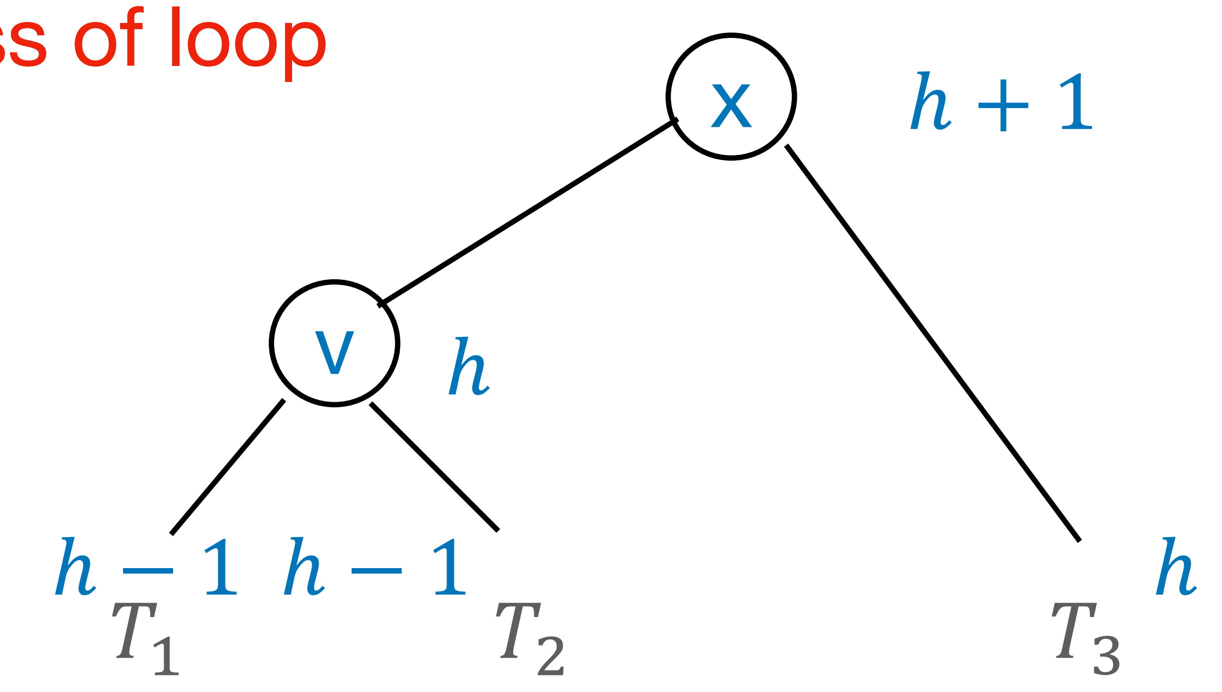
deletion in T_1



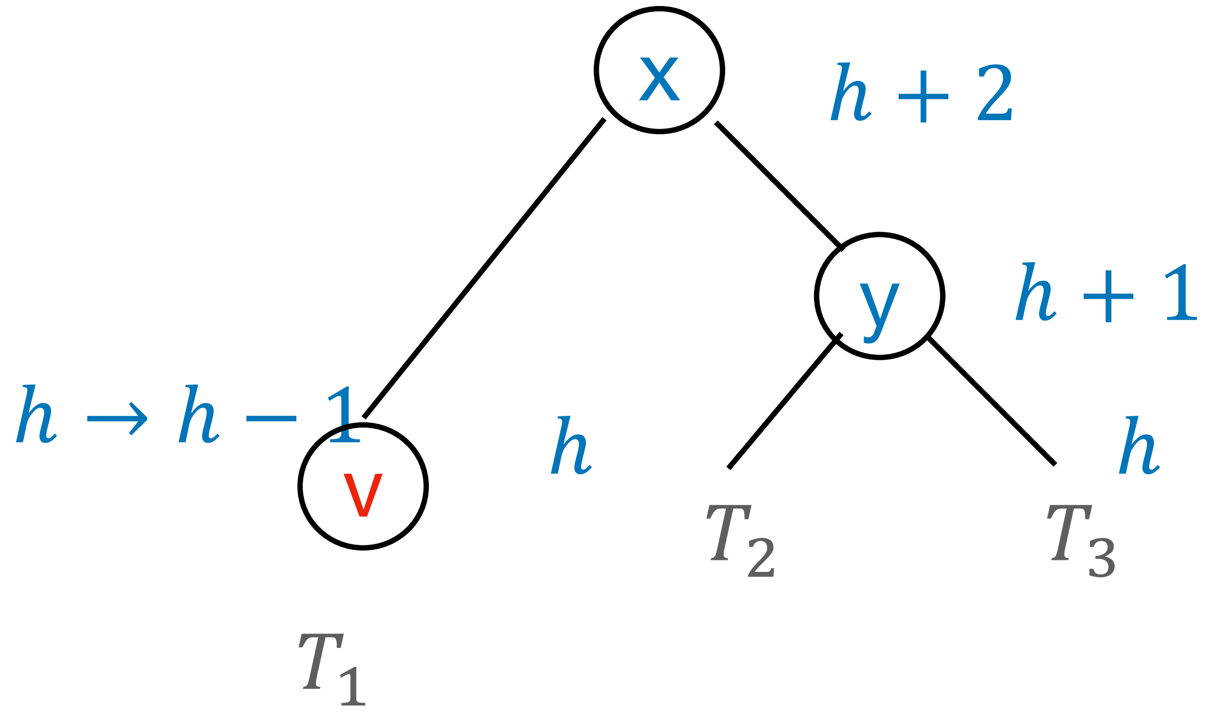
rotate left



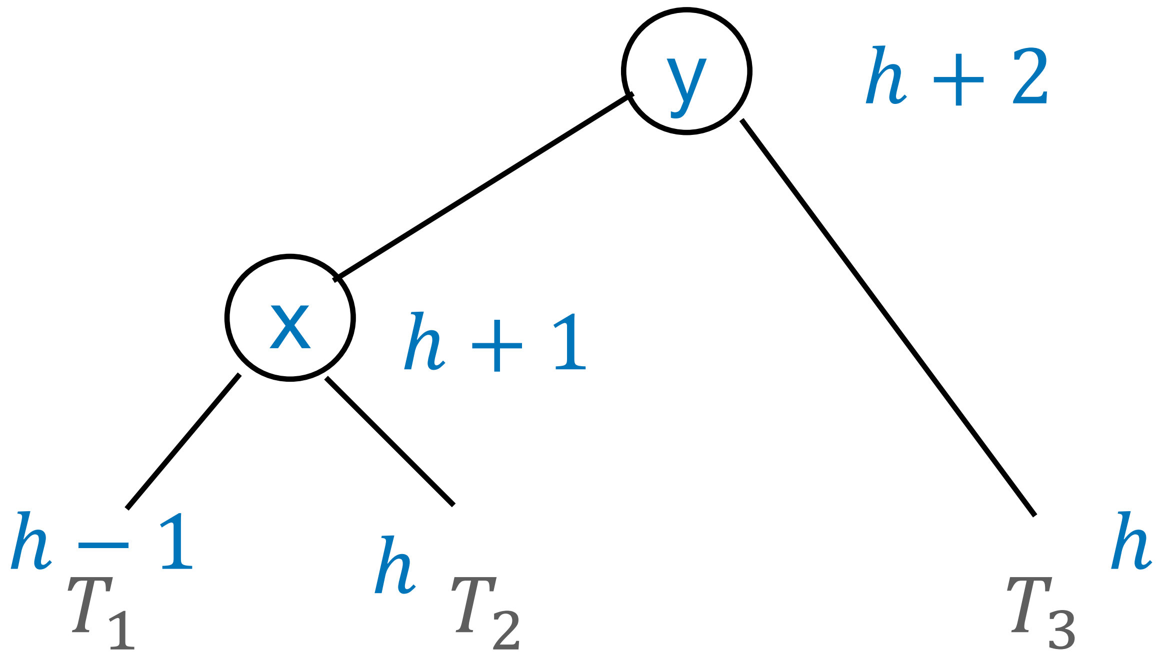
next pass of loop



- case $bal'(y) = 0$



rotate left



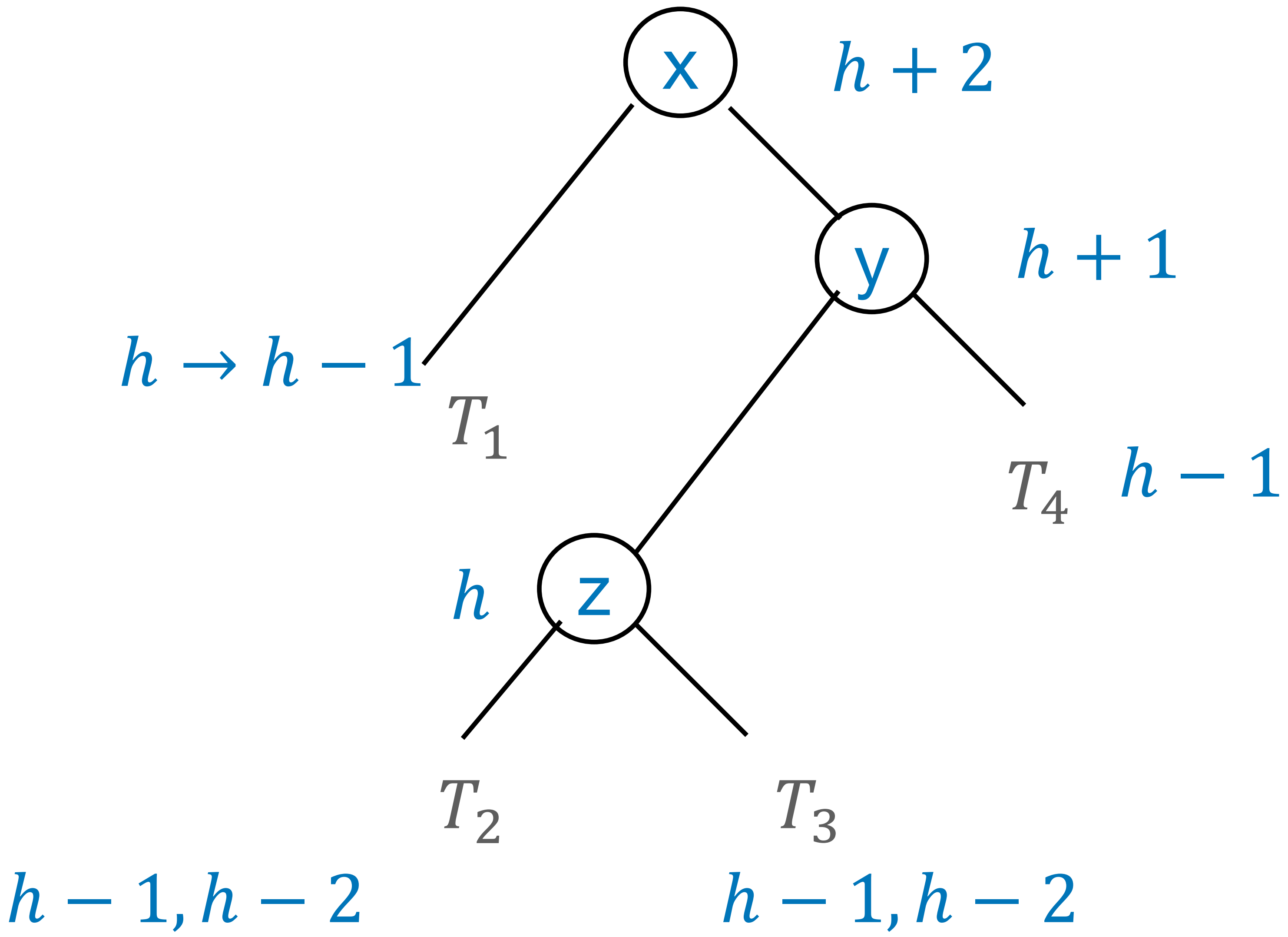
stop

correctness

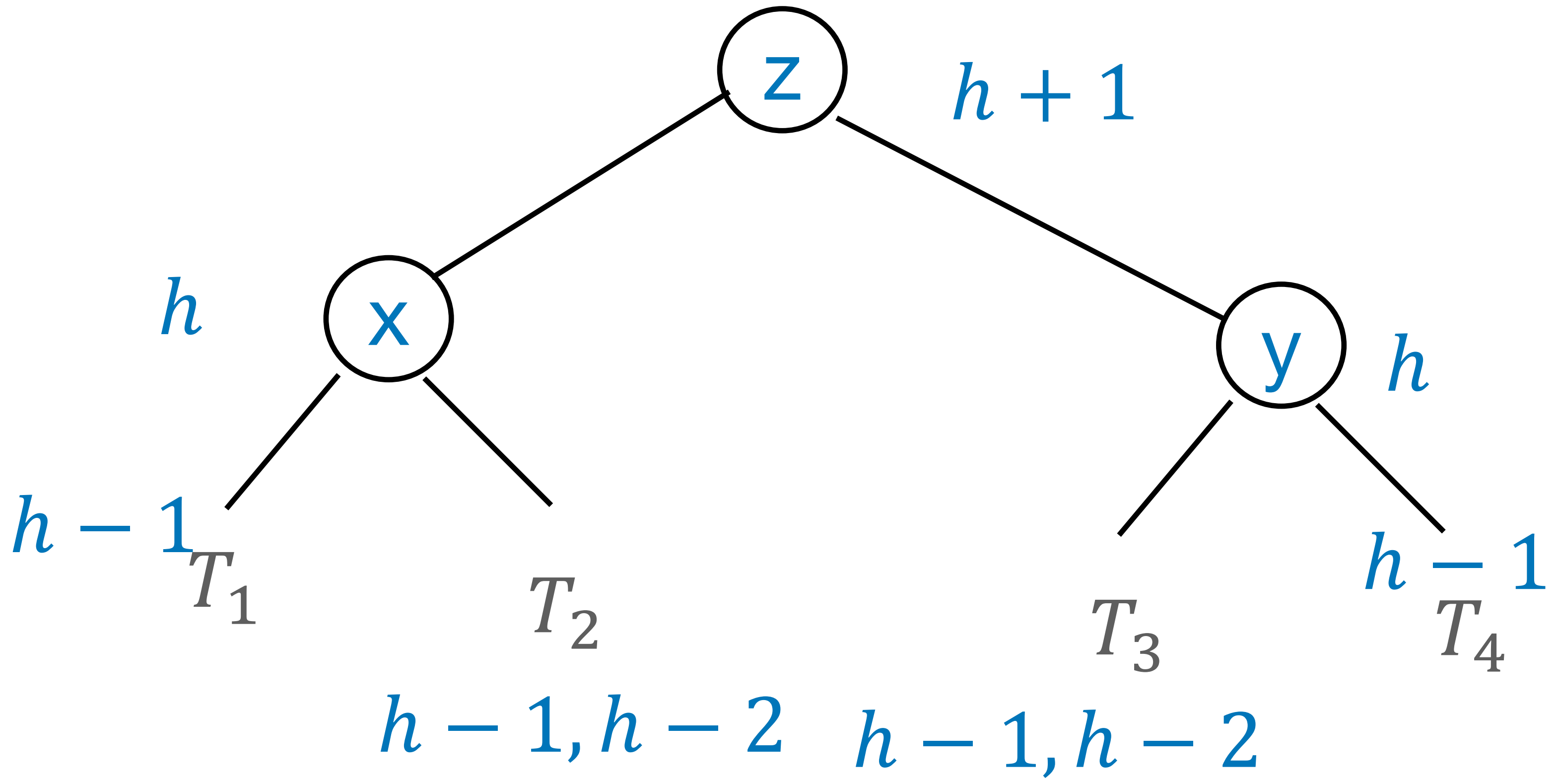
- here only $bal'(x) = -2$
  - case $bal'(y) = 1$

    deletion in T_1

to show:
rotations restore AVL property for $bal'(x) \in \{-2,2\}$



double rotate



correctness

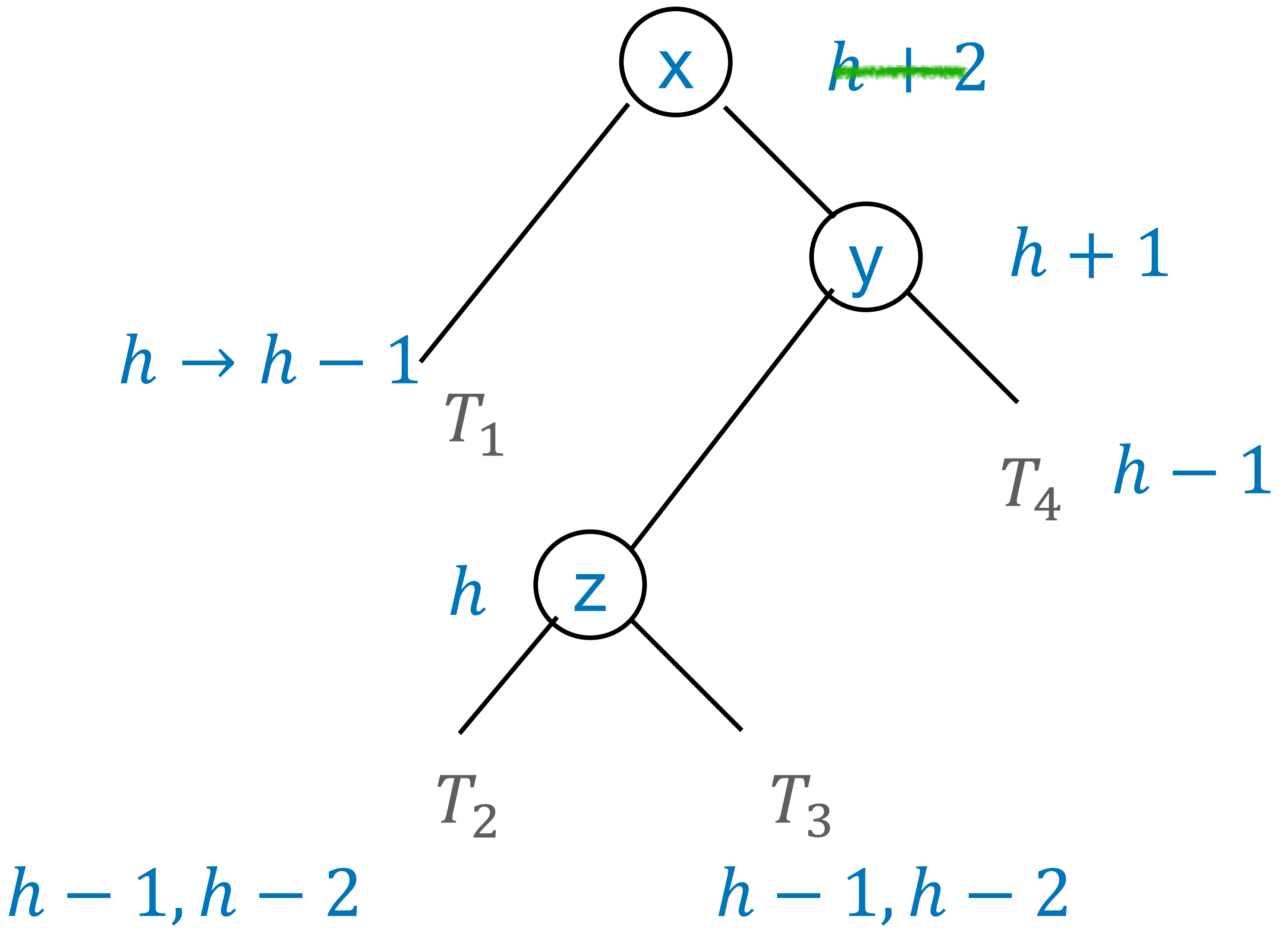
- here only $bal'(x) = -2$
  - case $bal'(y) = 1$

    deletion in T_1

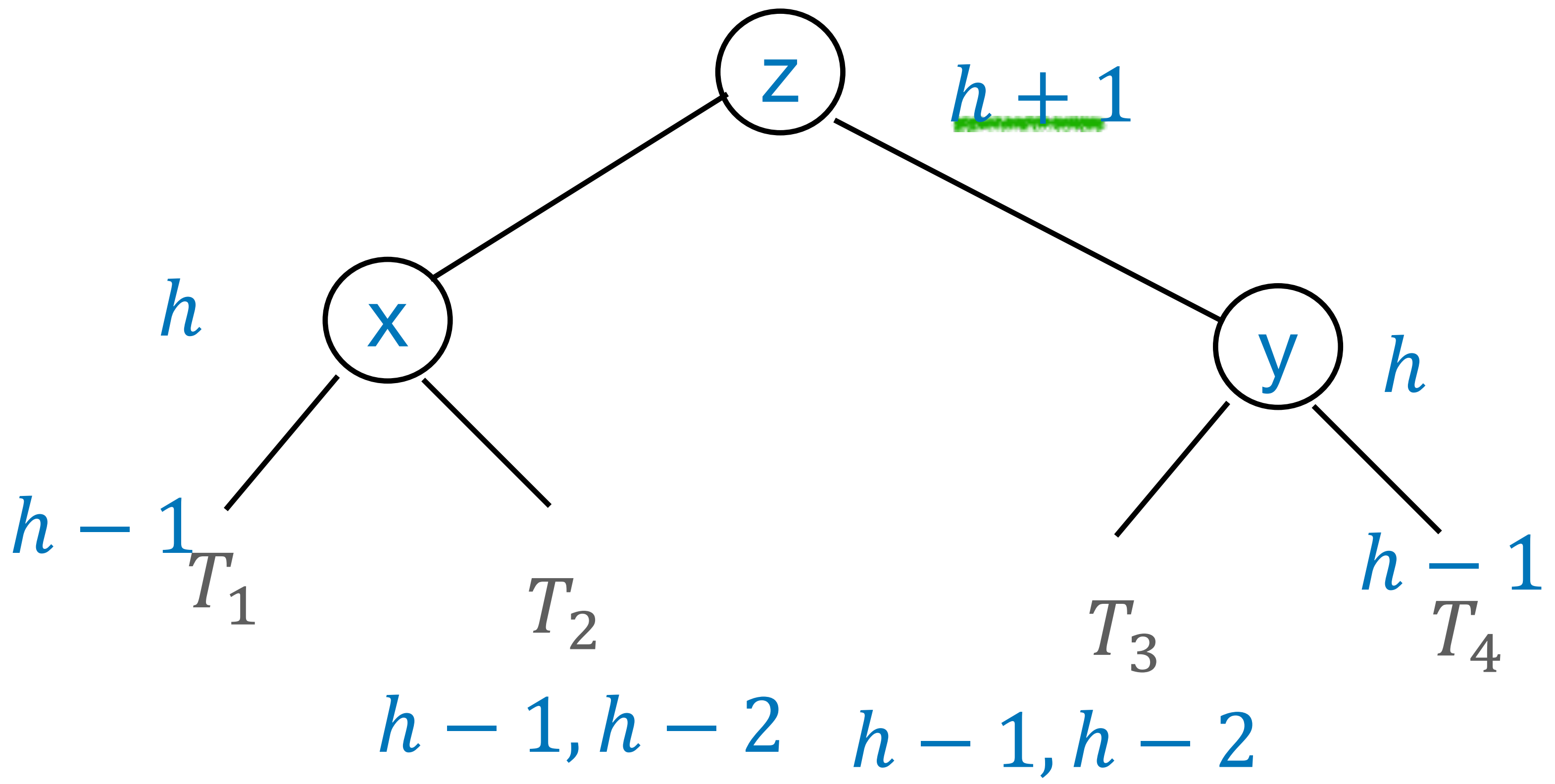
to show:

rotations restore AVL property for $bal'(x) \in \{-2,2\}$

height decreased



double rotate



next pass of loop