10. AVL Trees

Balanced Trees [Ottman/Widmayer, Kap. 5.2-5.2.1, Cormen et al, Kap. Problem 13-3]

Objective

Searching, insertion and removal of a key in a tree generated from n keys inserted in random order takes expected number of steps $\mathcal{O}(\log_2 n)$.

But worst case $\Theta(n)$ (degenerated tree).

Goal: avoidance of degeneration. Artificial balancing of the tree for each update-operation of a tree.

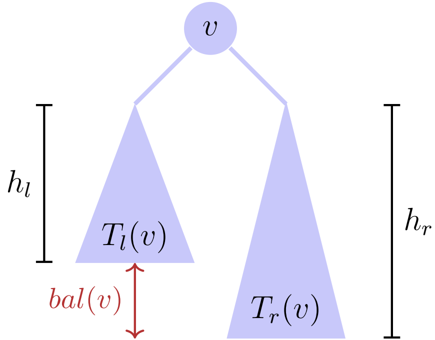
Balancing: guarantee that a tree with n nodes always has a height of $\mathcal{O}(\log n)$.

Adelson-Venskii and Landis (1962): AVL-Trees

Balance of a node

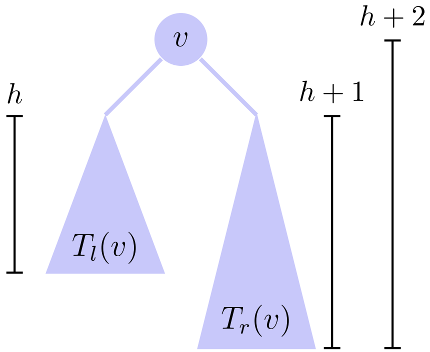
The height **balance** of a node v is defined as the height difference of its sub-trees $T_l(v)$ and $T_r(v)$

$$\mathrm{bal}(v) := h(T_r(v)) - h(T_l(v))$$

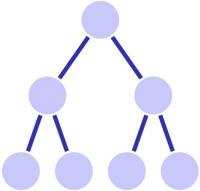


AVL Condition

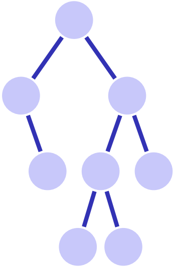
AVL Condition: for eacn node v of a tree
$\mathrm{bal}(v) \in \{-1, 0, 1\}$



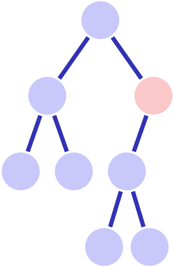
(Counter-)Examples



AVL tree with height 2



AVL tree with height 3

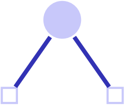


No AVL tree

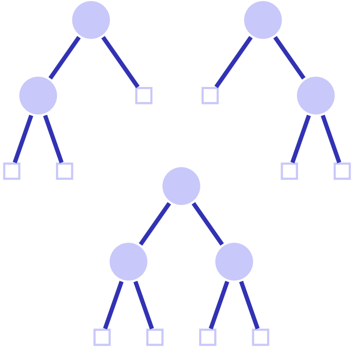
Number of Leaves

- 1. observation: a binary search tree with n keys provides exactly $n+1$ leaves. Simple induction argument.
  - The binary search tree with $n=0$ keys has $m=1$ leaves
  - When a key is added ($n \to n+1$), then it replaces a leaf and adds two new leafs ($m \to m-1+2=m+1$).
- 2. observation: a lower bound of the number of leaves in a search tree with given height implies an upper bound of the height of a search tree with given number of keys.

Lower bound of the leaves



AVL tree with height 1 has $N(1) := 2$ leaves.



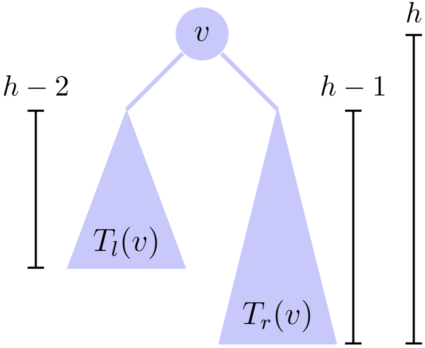
AVL tree with height 2 has at least $N(2) := 3$ leaves.

Lower bound of the leaves for $h > 2$

- Height of one subtree $\geq h - 1$.
- Height of the other subtree $\geq h - 2$.

Minimal number of leaves $N(h)$ is

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



Overal we have $N(h) = F_{h+2}$ with **Fibonacci-numbers** $F_0 := 0$, $F_1 := 1$, $F_n := F_{n-1} + F_{n-2}$ for $n > 1$.

Fibonacci Numbers, closed Form

It holds that

$$F_i = \frac{1}{\sqrt{5}}(\phi^i - \hat{\phi}^i)$$

with the roots $\phi, \hat{\phi}$ of the golden ratio equation $x^2 - x - 1 = 0$:

$$\phi = \frac{1+\sqrt{5}}{2} \approx 1.618$$
$$\hat{\phi} = \frac{1-\sqrt{5}}{2} \approx -0.618$$

Fibonacci Numbers, Inductive Proof

$F_i \overset{!}{=} \frac{1}{\sqrt{5}}(\phi^i - \hat{\phi}^i) \qquad [*]$ $\qquad \left(\phi = \frac{1+\sqrt{5}}{2}, \hat{\phi} = \frac{1-\sqrt{5}}{2}\right)$.

1. Immediate for $i = 0, i = 1$.
2. Let $i > 2$ and claim $[*]$ true for all F_j, $j < i$.

$$F_i \overset{def}{=} F_{i-1} + F_{i-2} \overset{[*]}{=} \frac{1}{\sqrt{5}}(\phi^{i-1} - \hat{\phi}^{i-1}) + \frac{1}{\sqrt{5}}(\phi^{i-2} - \hat{\phi}^{i-2})$$

$$= \frac{1}{\sqrt{5}}(\phi^{i-1} + \phi^{i-2}) - \frac{1}{\sqrt{5}}(\hat{\phi}^{i-1} + \hat{\phi}^{i-2}) = \frac{1}{\sqrt{5}}\phi^{i-2}(\phi + 1) - \frac{1}{\sqrt{5}}\hat{\phi}^{i-2}(\hat{\phi} + 1)$$

($\phi, \hat{\phi}$ fulfil $x + 1 = x^2$)

$$= \frac{1}{\sqrt{5}}\phi^{i-2}(\phi^2) - \frac{1}{\sqrt{5}}\hat{\phi}^{i-2}(\hat{\phi}^2) = \frac{1}{\sqrt{5}}(\phi^i - \hat{\phi}^i).$$

Tree Height

Because $|\hat{\phi}| < 1$, overal we have

$$N(h) \in \Theta\left(\left(\frac{1+\sqrt{5}}{2}\right)^h\right) \subseteq \Omega(1.618^h)$$

and thus

$$N(h) \geq c \cdot 1.618^h$$
$$\Rightarrow \quad h \leq 1.44 \log_2 n + c'.$$

An AVL tree is asymptotically not more than 44% higher than a perfectly balanced tree.[5]

[5] The perfectly balanced tree has a height of $\lceil \log_2 n + 1 \rceil$

Insertion

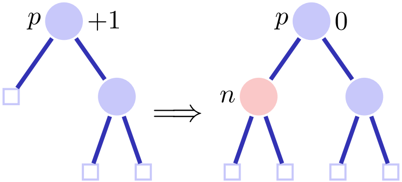
Balance

- Keep the balance stored in each node
- Re-balance the tree in each update-operation

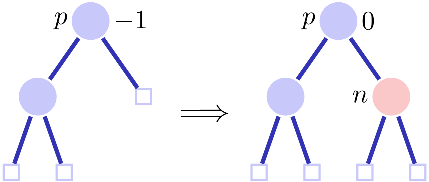
New node n is inserted:

- Insert the node as for a search tree.
- Check the balance condition increasing from n to the root.

Balance at Insertion Point



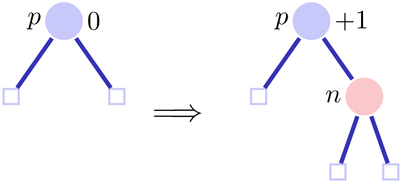
case 1: $\mathrm{bal}(p) = +1$



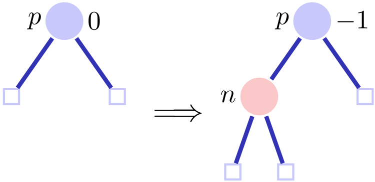
case 2: $\mathrm{bal}(p) = -1$

Finished in both cases because the subtree height did not change

Balance at Insertion Point



case 3.1: $\mathrm{bal}(p) = 0$ right



case 3.2: $\mathrm{bal}(p) = 0$, left

Not finished in both case. Call of **`upin(p)`**

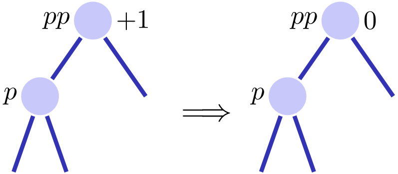
upin(p) - invariant

When **`upin(p)`** is called it holds that

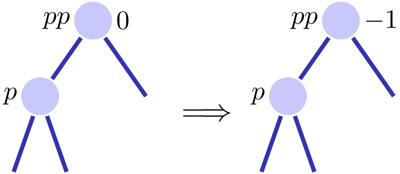
- the subtree from p is grown and
- $\text{bal}(p) \in \{-1, +1\}$

upin(p)

Assumption: p is left son of pp[6]



case 1: $\text{bal}(pp) = +1$, done.



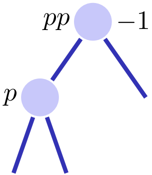
case 2: $\text{bal}(pp) = 0$, **`upin(pp)`**

In both cases the AVL-Condition holds for the subtree from pp

[6] If p is a right son: symmetric cases with exchange of $+1$ and -1

upin(p)

Assumption: p is left son of pp



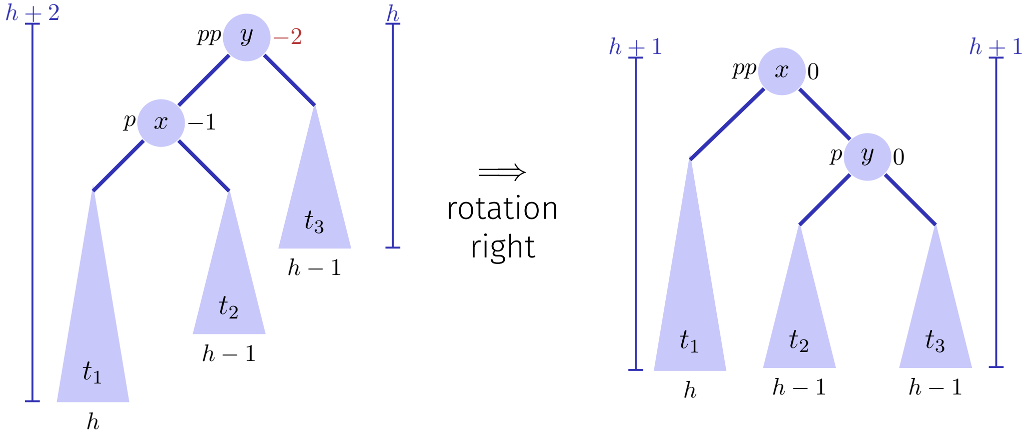
case 3: $\mathrm{bal}(pp) = -1$,

This case is problematic: adding n to the subtree from pp has violated the AVL-condition. Re-balance!

Two cases $\mathrm{bal}(p) = -1$, $\mathrm{bal}(p) = +1$

Rotations

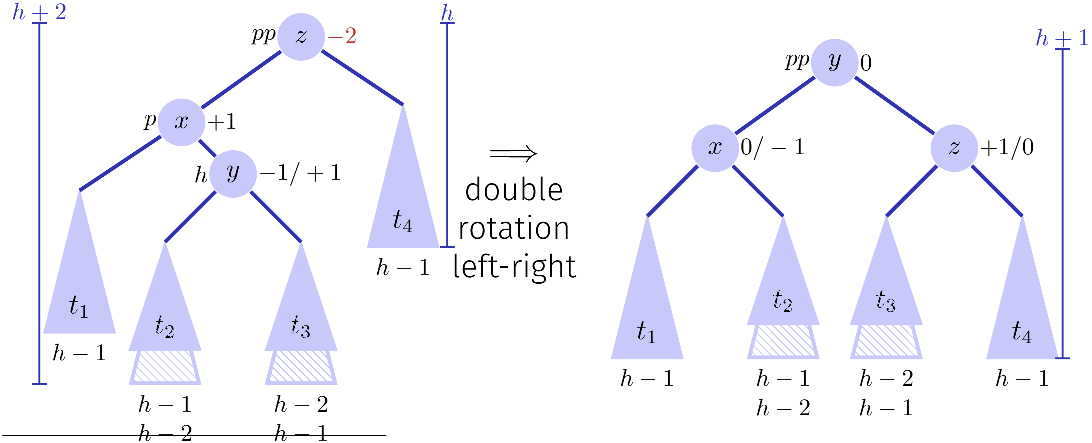
case 1.1 $\text{bal}(p) = -1$. [7]



[7] p right son: $\Rightarrow \text{bal}(pp) = \text{bal}(p) = +1$, left rotation

Rotations

case 1.1 $\text{bal}(p) = -1$.[8]



[8] p right son $\Rightarrow \text{bal}(pp) = +1$, $\text{bal}(p) = -1$, double rotation right left

Analysis

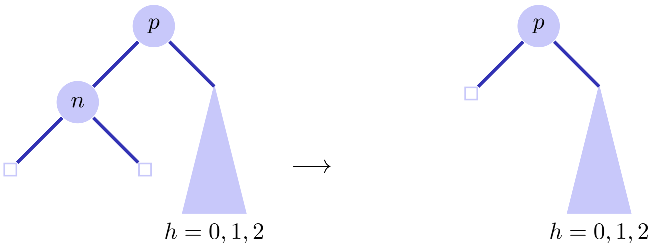
- Tree height: $\mathcal{O}(\log n)$.
- Insertion like in binary search tree.
- Balancing via recursion from node to the root. Maximal path lenght $\mathcal{O}(\log n)$.

Insertion in an AVL-tree provides run time costs of $\mathcal{O}(\log n)$.

Deletion

Case 1: Children of node n are both leaves Let p be parent node of n. $\Rightarrow$ Other subtree has height $h' = 0$, 1 or 2.

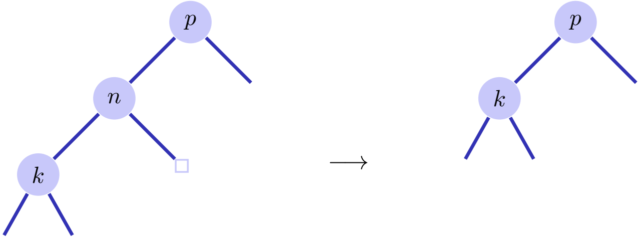
- $h' = 1$: Adapt $\text{bal}(p)$.
- $h' = 0$: Adapt $\text{bal}(p)$. Call **`upout(p)`**.
- $h' = 2$: Rebalanciere des Teilbaumes. Call **`upout(p)`**.



Deletion

Case 2: one child k of node n is an inner node

- Replace n by k. **`upout(k)`**



Deletion

Case 3: both children of node n are inner nodes

- Replace n by symmetric successor. **`upout(k)`**
- Deletion of the symmetric successor is as in case 1 or 2.

`upout(p)`

Let pp be the parent node of p.

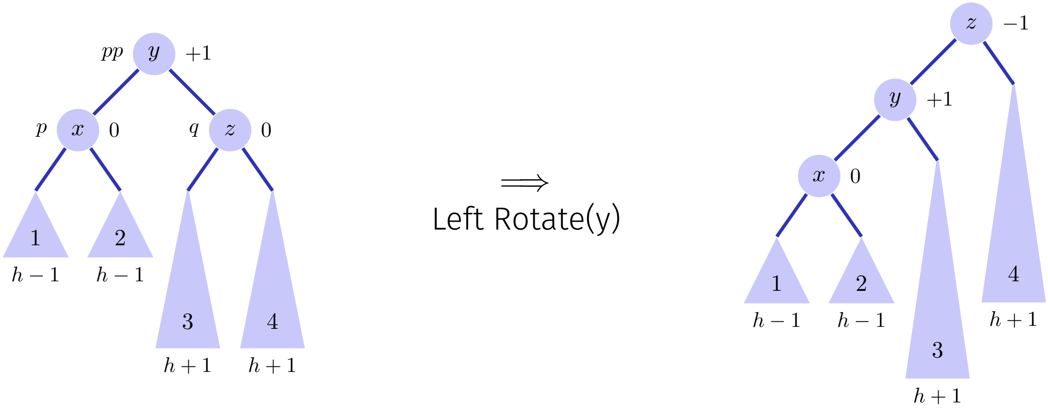
(a) p left child of pp

1. $\text{bal}(pp) = -1 \;\Rightarrow\; \text{bal}(pp) \leftarrow 0$. **`upout(pp)`**
2. $\text{bal}(pp) = 0 \;\Rightarrow\; \text{bal}(pp) \leftarrow +1$.
3. $\text{bal}(pp) = +1 \Rightarrow$ next slides.

(b) p right child of pp: Symmetric cases exchanging $+1$ and -1.

upout(p)

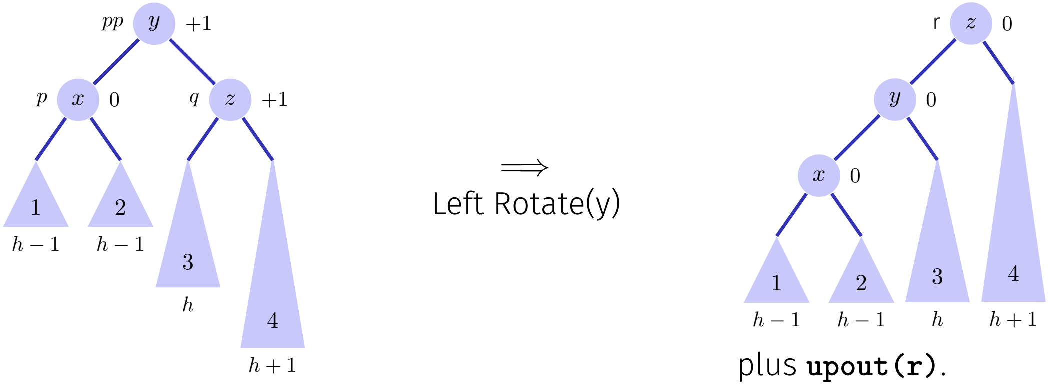
Case (a).3: $\text{bal}(pp) = +1$. Let q be brother of p
(a).3.1: $\text{bal}(q) = 0$.[9]



[9] (b).3.1: $\text{bal}(pp) = -1$, $\text{bal}(q) = -1$, Right rotation

`upout(p)`

Case (a).3: $\text{bal}(pp) = +1$. (a).3.2: $\text{bal}(q) = +1$.[10]

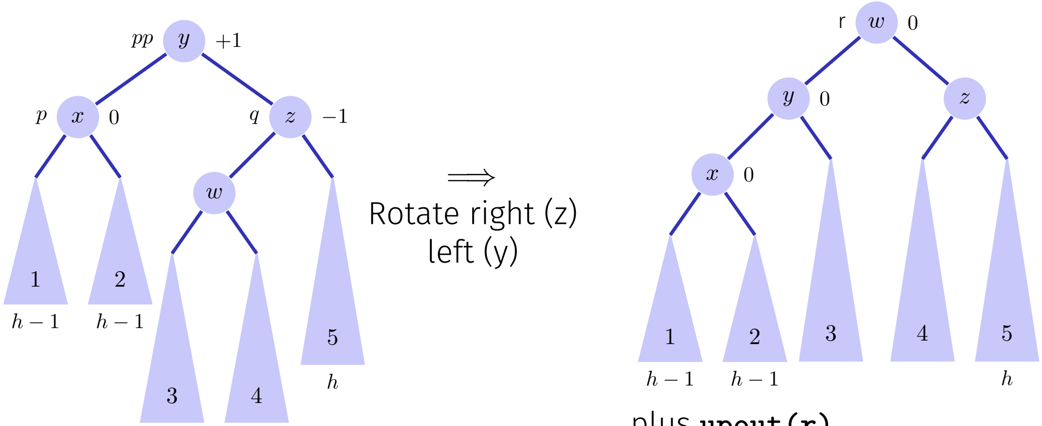


plus `upout(r)`.

[10](b).3.2: $\text{bal}(pp) = -1$, $\text{bal}(q) = +1$, Right rotation+upout

upout(p)

Case (a).3: $\mathrm{bal}(pp) = +1$. (a).3.3: $\mathrm{bal}(q) = -1$.[11]



plus **upout(r)**.

[11](b).3.3: $\mathrm{bal}(pp) = -1$, $\mathrm{bal}(q) = -1$, left-right rotation + upout

Conclusion

- AVL trees have worst-case asymptotic runtimes of $\mathcal{O}(\log n)$ for searching, insertion and deletion of keys.
- Insertion and deletion is relatively involved and an overkill for really small problems.