# Height-balanced tutorial:AVL[1]

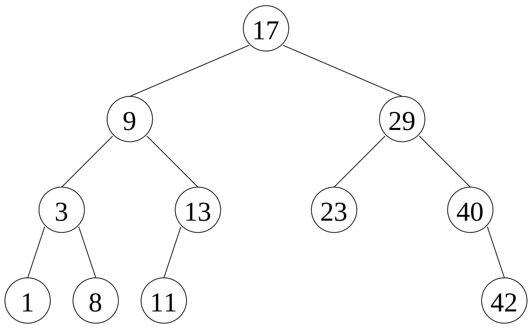

Figure 1: AVL ?

## 1 Preliminaries

**Exercise 1.1 (Balance factor (Déséquilibre))**

1. (a) What is the *balance factor* of a node?
   (b) Indicate on the tree in figure 1 the balance factors of all the nodes.
2. Is the tree in figure 1 an AVL? Why?

LIFE IS ABOUT BALANCE AND STAYING QUICK AND ALERT AS EVERYTHING THREATENS TO SPIN OUT OF CONTROL!

**Exercise 1.2 (Height-balanced (H-équilibré)?)**

Write a function that tests whether a binary tree (class `BinTree`) is height-balanced.

[1]Adelson-Velskii & Landis

## 2 Rotations

**Exercise 2.1 (Rotations and balance factors)**

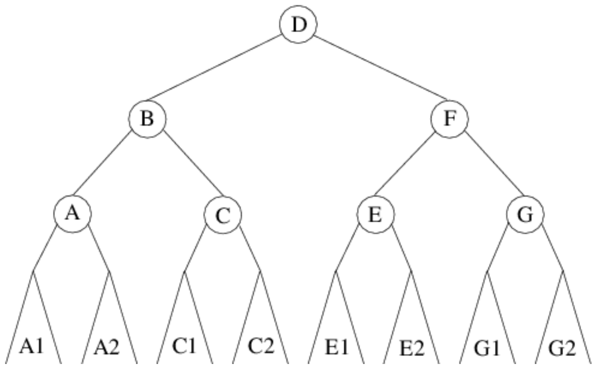

Figure 2: AVL: *Circles are nodes and triangles subtrees*

The figure 2 represents the structure of an AVL of height $\geq 2$.

1. The aim here is to determine which rotation has to be used in each unbalanced case, and the new balance factors after each rotation.
   (a) **Left rotation:**
      i. Draw the tree obtained after a left rotation on $D$.
      ii. Which nodes had their balance factor change?
      iii. For each unbalanced case, give the old and the new balance factors of each concerned node after the left rotation. Fill the table give in appendix.
      iv. For which cases is the left rotation rebalancing the tree ?
   (b) Considering only "interesting" cases, study the **right-left rotation** (see appendix) using the same methodology.

   **Fill the tables given in the appendix (last page). The last table has to summarize the cases where rotations are useful for re-balancing.**

2. To simplify insertion and deletion algorithms, it is interesting to integrate the balance factor updates to the rotation algorithms.
   (a) Why can double rotations no longer be implemented using single rotations?
   (b) Given that the rotations will only be used in the cases considered here (precise specifications should be given for each rotation) write the four rotation functions including the balance factor updates.

---

**Exercise 2.2 (Rotations and heights)**

After a rotation, the height of the given tree may change. It is necessary to know in which cases the height changes to indicate to the ancestor nodes that one of the height of its sub-trees has changed (which modifies the balance factor of the ancestor nodes).

For each rotation, use the trees obtained in the previous exercise and answer the following questions:

1. In which cases did the height of the tree change?
2. Add height variations for each case to **the table 3 given in appendix**.

## 3 Modifications

**Exercise 3.1 (Insertion)**

A recursive version of the element insertion in an AVL will be written in this exercise. It will be based on the principle of insertion at the leaf in binary search trees, with the rebalancing "in going up".

1. After the insertion, some subtree heights may have changed. In order to be able to update the balance factors, this information has to be pulled up on the path from the new leaf to the root.
   (a) Considering an insertion in the left subtree, which has increased the height of this subtree, consider the different cases depending on whether the starting tree has a balance factor of -1, 0 or 1 (see figures 3 to 5). Some cases should be detailed (make "sub-cases").

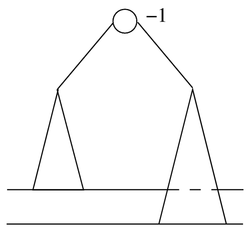

Figure 3: Balance factor: -1

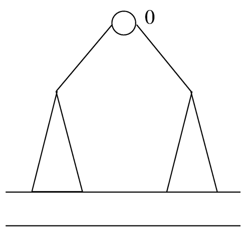

Figure 4: Balance factor: 0

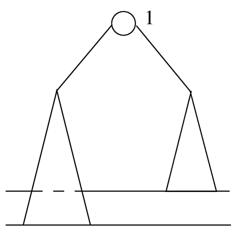

Figure 5: Balance factor: 1

   (b) Deduce, from balance factor before insertion ($bal_i$), after insertion ($bal_f$), after rotation ($bal_r$), the tree height changes.
2. Deduce the principle of the insertion in an AVL which separates the "insertion" and the "rebalancing".
3. Add the keys 4, 48, 7, 5 and 6 in the tree in figure 1.
4. Write the insertion function.

---

**Exercise 3.2 (Deletion)**

The deletion will be done on the same model as the insertion:

- The deletion itself will be done as on the same model as the one seen in tutorial for binary search trees.
- Rebalacing will be made in going up.

1. Study, using the same methodology as the insertion study, the height differences induced after deletion and possible rebalancing in the left subtree (resume figures 3 to 5).
2. Deduce the principle of the deletion in an AVL which separates the "deletion" and the "rebalancing".
3. Delete the keys 23, 17, 11 and 1 in the tree in figure 1.
4. Write the deletion function.

## Tables to fill

### Rotations and balance factors

| left rotation (lr) | | | |
|---|---|---|---|
| **bal(D)** | **bal(F)** | **bal'(D)** | **bal'(F)** |
| -2 | -1 | | |
| | 0 | | |
| | +1 | | |
| +2 | -1 | | |
| | 0 | | |
| | +1 | | |

Table 1: New balance factor values after left rotation

| right-left rotation (rlr) | | | | | |
|---|---|---|---|---|---|
| **bal(D)** | **bal(F)** | **bal(E)** | **bal'(D)** | **bal'(F)** | **bal'(E)** |
| | | | | | |
| | | | | | |
| | | | | | |

Table 2: New balance factor values after right-left rotation ("useful" cases)

### Summary: rotations and height changes

| **bal(root)** | ***bal(left child)*** ***bal(right child)***[1] | **rotation** | $\Delta H$ |
|---|---|---|---|
| | | left | |
| | | | |
| | | right-left | |
| **bal(root)** | ***bal(left child)*** ***bal(right child)***[1] | **rotation** | $\Delta H$ |
| | | left-right | |
| | | right | |
| | | | |

[1] *Delete as appropriate*

Table 3: Height changes