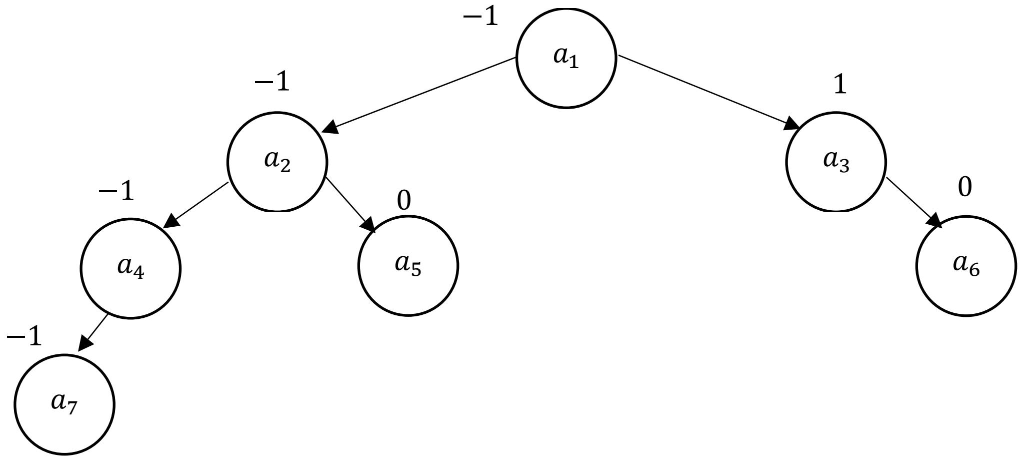
ADT AVL Tree

AVL Tree = $\{a_1, a_2, a_3, a_4, \ldots a_n\}$

Where a_1 is the root of the tree, the elements a_2 and a_3 are the children of a_1, and in addition to this, they are also subtrees, that they have children. In addition, each node has a roll factor, which is obtained by subtracting the height of the right child of a node by the height of the left child of the same node.

Graphic Representation



$a_7 < a_4 < a_2 < a_2 < a_1 < a_3 < a_6$

Note: The roll factor will be represented by $fb(x)$

$$inv = \{a_1 > a_2, a_1 < a_3 \rightarrow \ a_2 < a_1 < a_3\}$$
$$inv = \{fb(x) = z \ \wedge \big(z \mid z \ \in (-1, 0, 1)\big)\}$$

Primitive Operations

AVLTree	…	AVLTree
Add	AVLTree x Key x Value	Node
Delete	AVLTree x Key x Value	Node
Left Rotate	AVLTree x Node	AVLTree
Right Rotate	AVLTree x Node	AVLTree
Get Balance Factor	AVLTree x Key x Value	Int
Rebalance	AVLTree x Node x Key	AVLTree
Max	AVLTree x Int x Int	Int
Height	AVLTree x Node	Int
Update Height	AVLTree x Node	AVLTree

AVLTree( ) : Constructor
Create the AVL Tree
$pre = \{true\}$ $pos = \{AVLTree\}$

Add(K key, V value) : Modifier
Add a new element in the AVLTree, if the new element's key is equals to another element, so the new element will be added in the left son
$pre = \{element\}$ $pos = \{new\ element\ in\ the\ AVLTree \mid root\ != null\}$

Delete(K key, V value): Modifier
Search for the corresponding node, and after deleting this node, but first the program must validate if the value is equals to the value of corresponding node
$pre = \{root\ != null \wedge k \in AVLTree\}$ $pos = \{new\ order\ of\ nodes\ and\ one\ less\ element\}$

Left Rotate(Node x) : Modifier
Rotate nodes to the right if the roll factor of node x is greater than 1.
$pre = \{node\ x\ with\ the\ factor\ balance\ \ greather\ than\ 1\}$ $pos = \{new\ factor\ balance\ in\ the\ node\ x\}$

Right Rotate (Node x) : Modifier
Rotate nodes to the left if the roll factor of node x is less than -1
$pre = \{node\ x\ with\ the\ factor\ balance\ \ less\ than -1\}$ $pos = \{new\ factor\ balance\ in\ the\ node\ x\}$

Get Balance Factor(Key k, V value) : Analyzer
Is the difference between the heigh of the right son for the height of the left son
$pre = \{true\}$ $pos = \{balance\ factor\ of\ the\ node\ with\ key\ k\ and\ value\ v\}$

Rebalance(Node x) : Modifier
Evaluate the factors balance of the nodes and call the methods right Rotate and left Rotate to rebalance the AVLTree
$pre = \{true\}$ $pos = \{new\ order\ of\ the\ nodes\ in\ the\ AVLTree\}$

Max(Int a, Int b) : Analyzer
Evaluate two integers to know what of these are the greater
$pre = \{true\}$ $pos = \{the\ integer\ greather\}$

Height(Node x): Analyzer
Get the heigh of the node x
$pre = \{true\}$ $pos = \{the\ height\ of\ the\ node\ x\ and\ a\ integer\}$

Update Height(Node x): Modifier
When a node is added, his ancestors have a new height
$pre = \{new\ node\ in\ the\ AVLTree\}$ $pos = \{new\ height\ of\ the\ nodes\ in\ the\ AVLTree\}$