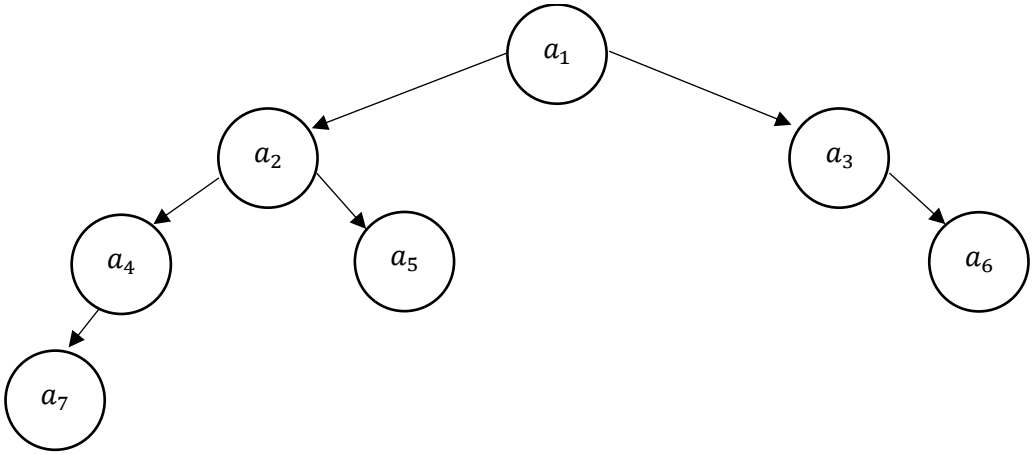
ADT Binary Search Tree

Binary Search Tree $= \{a_1, a_2, a_3, a_4, \dots, a_n\}$

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

Graphic Representation



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

$$inv = \{a_1 > a_2, a_1 < a_3 \rightarrow a_2 < a_1 < a_3\}$$

Primitive Operations

Name	Input	Output
BinarySearchTree	…	BinarySearchTree
InOrder	BinarySearchTree	String
Search	BinarySearchTree x Key	Node (any a)
Minimum	BinarySearchTree x Key	Node (the smallest a)
Maximum	BinarySearchTree x Key	Node (the greatest a)
Successor	BinarySearchTree x Key	Node (the largest of those smaller than a)
Insertion	BinarySearchTree x Key x Value	BinarySearchTree
Delete	BinarySearchTree x Key	BinarySearchTree

BinarySearchTree( ): Constructor
Create the binary tree by initializing the root
$pre = \{true\}$ $pos = \{root\ initialized\}$

InOrder( ): Analyzer
Traversing the binary search tree in order
$pre = \{root\ != null\}$ $pos = \{BinarySearchTree\}$

Search(Key k): Analyzer
Searches for an element passed by parameter in the binary search tree
$pre = \{root\ != null\ \wedge k\ \in\ tree\ \}$ $pos = \{a\ node\ with\ key\ k\}$

Minimum(Key k): Analyzer
Gets the smallest element in the binary search tree according to the value of k
$pre = \{root\ != null\ \wedge\ k\ \in\ tree\}$ $pos = \{the\ smallest\ node\}$

Maximum(Key k): Analyzer
Gets the largest element in the binary search tree according to the value of k
$pre = \{root\ != null\ \wedge\ k\ \in\ tree\}$ $pos = \{the\ largest\ node\ \}$

Successor(Key k): Analyzer
Gets the smallest element in the set of elements greater than the key k
$pre = \{root\ != null\ \wedge k\ \in tree\}$ $pos = \{the\ successor\ of\ k\}$

Insertion(Key k, Value v): Modifier
Adds a node to the binary search tree, this node is added according to the order criterion established in the invariant
$pre = \{true\}$ $pos = \{true, the\ nodo\ was\ added\}$

Delete (Key k): Modifier
Deletes an element belonging to the binary search tree, then re-sorts the tree
$pre = \{root\ != null\ \wedge\ k \in in\ the\ tree\}$ $pos = \{new\ order\ of\ nodes\ and\ one\ less\ element\}$