**Juan José Barrera Gracia – A00394876**

**Computación y estructuras discretas I**

**Seguimiento III Generics y TAD**

**Universidad ICESI**

**Andrés Aristizábal**

**2023-1**

**Tras analizar la estructura de datos, se determinó que la mejor forma de realizar su estudio es mediante dos TAD’s, El TAD del nodo y el TAD del árbol binario. A continuación, se presentan ambos análisis.**

|  |
| --- |
| **TAD: Node** |
| **Node: {NodeRight:<nodeRight>,NodeLeft:<nodeLeft>,T:<value>,K:<key>}** |
| **{Inv: Node.Right>Node && Node.Left<Node && Key extends comparable}** |
| **Operaciones Primitivas:**  **Constructoras:**  createNode: Value X Key 🡪 Node  **Modificadoras:**  setValue: Node X Value 🡪 Node  setKey: Node X Key 🡪 Node  setRight: NodeXNodeRight 🡪 NodeRight  setLeft: NodeXNodeLeft🡪 NodeLeft  **Analizadoras:**  getValue: Node 🡪 Value  getKey: Node🡪 Key  getRight: Node🡪 NodeRight  getLeft: Node🡪 NodeLeft |

|  |
| --- |
| **setKey(node, key):** |
| “Set the key of a node” |
| {Pre: Node !=null && K extends comparable} |
| {Pos: node.key=key} |

|  |
| --- |
| **setValue(node, value):** |
| “Set the value of a node” |
| {Pre: node != null} |
| {Pos: node.value=value} |

|  |
| --- |
| **createNode(key, value):** |
| “Create a Node using a key and a value” |
| {Pre: key extends comparable} |
| {Pos: node = {T,K}} |

|  |
| --- |
| **setRight(node, nodeRight):** |
| “Set the right of a node” |
| {Pre: node&&node<node.right} |
| {Pos: node.right=nodeRight} |

|  |
| --- |
| **setLeft(node, nodeLeft):** |
| “Set the left of a node” |
| {Pre: node&&node>node.left} |
| {Pos: node.left=nodeLeft} |

|  |
| --- |
| **getValue(node):** |
| “Get the value of a node” |
| {Pre: node&&node != null} |
| {Pos: <value>} |

|  |
| --- |
| **getKey(node):** |
| “Get the value of a node” |
| {Pre: node&&node!=null} |
| {Pos: <key>} |

|  |
| --- |
| **getLeft(node):** |
| “Get the left of a node” |
| {Pre: node&&node!=null  &&node.Left!=null} |
| {Pos: <node.Left>} |

|  |
| --- |
| **getRight(node):** |
| “Get the right of a node” |
| {Pre: node&&node!=null&&  node.Right!=null} |
| {Pos: <node.Right>} |

|  |
| --- |
| **TAD: BinarySearchTree** |
| **BinarySearchTree={Root:<root>}** |
| **{Inv: root != null}** |
| **Operaciones primitivas:**  **Modificadoras:**  Insert: TreeXNode🡪 Tree  delete: TreeXKey 🡪 Tree  **Analizadoras:**  getRoot: Tree 🡪 Value  search: TreeXKey🡪 Value  InOrder:Tree🡪 String |

|  |
| --- |
| **getRoot(tree):** |
| “Get the root of a tree” |
| {Pre: root != null} |
| {Pos: <root>} |

|  |
| --- |
| **delete(tree, key):** |
| “Delete a node of the tree” |
| {Pre: tree && tree.node != null} |
| {Pos: tree.node = null} |

|  |
| --- |
| **insert(tree, node):** |
| “Insert a new node in the tree” |
| {Pre: true} |
| {Pos: tree = {node}} |

|  |
| --- |
| **search(tree, key)** |
| “Search a node in a tree” |
| {Pre: tree && tree.node!=null} |
| {Pos: <tree.node>} |

|  |
| --- |
| **inOrder(tree)** |
| “Prints the tree in order” |
| {Pre: true} |
| {Pos:<tree.toString>} |