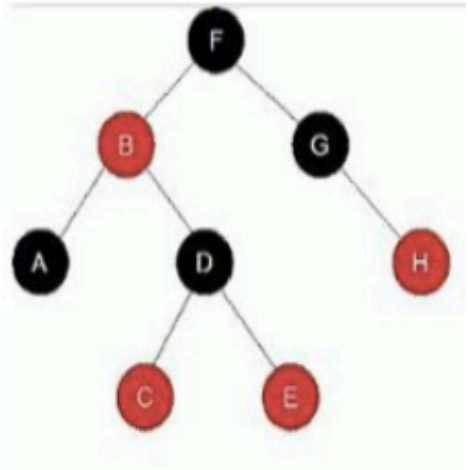
TAD Árbol Roji-negros




{invariante: $G \geq F \ \wedge \ B \leq F$ - Cada nodo es rojo o negro. - Toda hoja (NIL) es negra. - Si un nodo es rojo, sus dos hijos son negros. - Todo camino desde un nodo a cualquier hoja descendente contiene el mismo número de nodos negros.}

Operaciones:

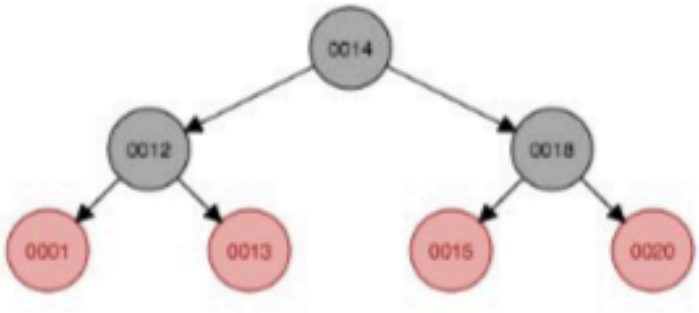
- crearArbolRJ: - →ArbolRoji-Negros (Constructora)
- agregar : ArbolRoji-Negros X Elemento → ArbolRoji-Negros (Modificadora)
- eliminar: ArbolRoji-Negros X Elemento → - (Destructora)
- girarIzquierda: ArbolRoji-Negros X Elemento → ArbolRoji-Negros (Modificadora)
- girarDerecha : ArbolRoji-Negros X Elemento → ArbolRoji-Negros (Modificadora)
- validar: ArbolRoji-Negros X Elemento → Boolean (Consultora)

eliminar(n)

"elimina el nodo que se pasa por parámetro"

{pre: n es de tipo nodo}

{post: ArbolRoji-Negros=




}

validar()

“Valida si el árbol se encuentra vacío ”

{pre: Árbol creado}

{post: True si la raiz es igual a null, false de lo contrario }

girarIzquierda(n)

“el árbol es girado hacia la izquierda sobre el nodo n ”

{pre: Árbol creado}

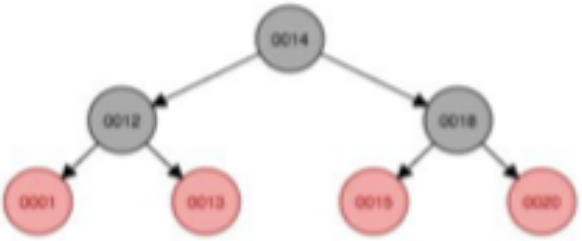
{post: el árbol es girado a la izquierda
}

crearArbolRoji-Negro()

“crea un nuevo árbol de tipo Roji-Negros”

{pre: True}

{post: Árbol  Roji-Negros=

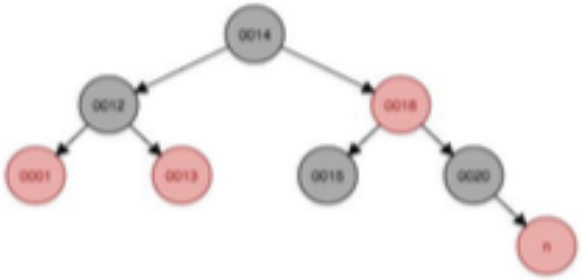



}

agregar(ArbolRoji-Negros, n)

“agrega un nodo al árbol Roji-Negros”

{pre: True, n es de tipo nodo}




{post: ArbolRoji-Negros=
}

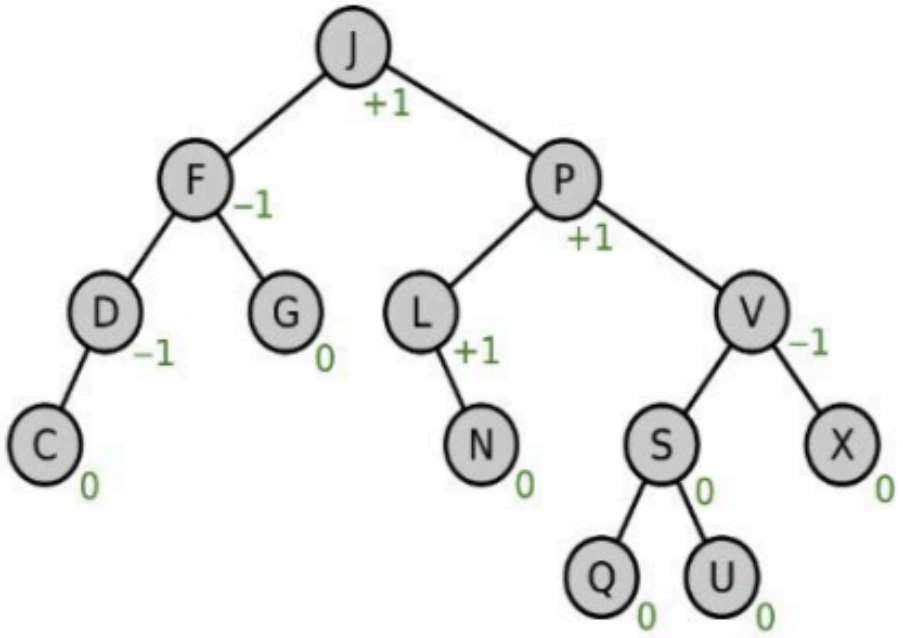
girarDerecha()

"el árbol es girado hacia la derecha sobre el nodo n "

{pre: Árbol creado}

{post: el árbol es girado hacia la derecha }

TAD Árbol AVL




{invariante: $P \geq J \ \wedge \ F \leq J \quad \wedge \ -2 \leq Factor\ de\ balance \leq 2$}

Operaciones:

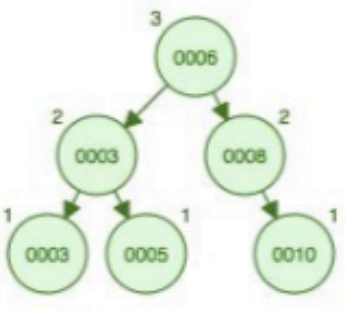
- crearArbolAVL: - →ArbolAVL (Constructora)
- agregar : ArbolAVL X Elemento → ArbolAVL (Modificadora)
- eliminar: ArbolAVL X Elemento → - (Destructora)
- validar: ArbolAVL X Elemento → Boolean (Consultora)
- girarIzquierda: ArbolAVL X Elemento → ArbolAVL (Modificadora)
- girarDerecha : ArbolAVL X Elemento → ArbolAVL (Modificadora)

crearArbolAVL()

“crea un nuevo árbol de tipo AVL”

{pre: True}

{post: ArbolAVL=



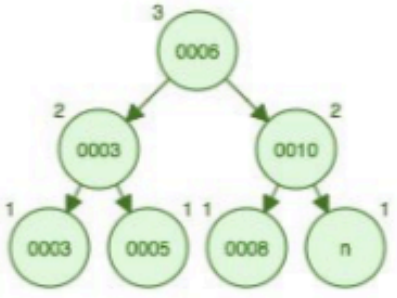

}

agregar(ArbolAVL, n)

“agrega un nodo al árbol AVL”

{pre: True, n es de tipo nodo}

{post: ArbolAVL=



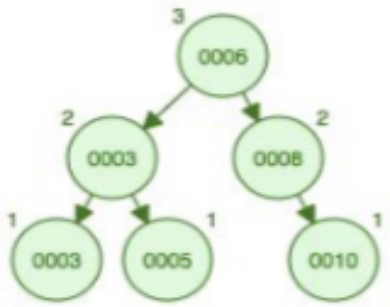

}

eliminar(n)

“elimina el nodo que se pasa por parámetro”

{pre:  n es de tipo nodo}

{post: ArbolAVL=




}

validar()

“Valida si el árbol se encuentra vacío ”

{pre: Árbol creado}

{post: true si la raíz es null, false de lo contrario
}

girarIzquierda(n)

“el árbol se gira a la izquierda sobre el nodo pasado por parámetro ”

{pre: Árbol creado}

{post:  el árbol es girado a la izquierda sobre el nodo}

girarDerecha(n)

“gira el árbol a la derecha sobre el nodo que pasa por parámetro ”

{pre: Árbol creado}

{post: el árbol es girado a la derecha sobre el nodo pasado por parámetro. }