Universidad Internacional de Valencia (VIU)

15GIIN – Estructuras de Datos y Algoritmos

Cuarto Portafolio – ACT4

Alumno: Gagliardo Miguel Angel

Ejercicio 1

Hacer un método de la clase **BinarySearchTree** que cuente las hojas de un árbol binario de búsqueda, usar la implementación de **BinarySearchTree** que se anexa a la actividad. Recuerde que una hoja es un nodo del árbol donde los apuntandores (referencias) a nodo izquierdo y derecho son nulos.

```
// Metodo privado de la clase para contar todas las hojas a partir de un nodo
private int contarHojas(BinaryNode nodo) {
	// Si el nodo es nulo, se devuelve 0
	if (nodo == null) {
		return 0;
	}
	// Si el nodo no tiene apuntadores (referencias) a nodo izquierdo ni
derecho, entonces es una hoja
	// Se devuelve 1
	if (nodo.left == null && nodo.right == null) {
		return 1;
	}
	else {
	// Llamada recursiva a contarHojas sumando los apuntadores (referencias)
del nodo izquierdo y derecho
		return contarHojas(nodo.left) + contarHojas(nodo.right);
	}
}
```

Ejercicio 2

Hacer un método menoresQue de la clase BinarySearchTree que cuente el número de elementos menores estrictos que un elemento dado, que llamamos ELEMENTO, en un árbol binario de búsqueda. ELEMENTO puede que no esté en el árbol.

```
 public int menoresQue(AnyType elemento) {
   // Si el arbol es vacio, la cantidad de elementos es 0
   if (this.isEmpty()) {
       return 0;
   }
   return this.menoresQue(elemento, this.getRoot());
 }

 private int menoresQue(AnyType elemento, BinaryNode root) {
   // Si la raiz con la cual comparamos elemento es nula, se devuelve un 0
   if (root == null) {
       return 0;
   }

   // Si elemento es igual al elemento raíz del árbol entonces contar el número de nodos del árbol izquierdo
   if (this.myCompare(elemento, (AnyType) root.getElement()) == 0) {
       return contarNodos(root.getLeft());
   }

   // Si elemento es menor al elemento raíz entonces recursivamente devolvemos el número de elementos menores
   // que el elemento izquierdo de la raiz del arbol que estamos analizando
     else if (this.myCompare(elemento, (AnyType) root.getElement()) < 0) {
       return menoresQue(elemento, root.getLeft());
   } else {
     // Finalmente: Si elemento es mayor que la raíz del árbol, el número de elementos menores que elemento serán 1 (por la
     // raíz) más número de nodos del árbol izquierdo, más el número que resulte de calcular recursivamente el número
     // de nodos menores que elemento en el árbol derecho
     return 1 + contarNodos(root.getLeft()) + menoresQue(elemento, root.getRight());
   }
 }

 // Metodo auxiliar que cuenta todos los nodos debajo de un nodo raiz, que es el nodo dado por parametro
 private int contarNodos(BinaryNode root) {
   // Caso base: El nodo raiz es nulo. Entonces se devuelve 0
   if (root == null) {
       return 0;
   }

   // Si el nodo raiz no es nulo, recursivamente sumo todos los nodos izquierdos y derechos del raiz
   return 1 + contarNodos(root.getLeft()) + contarNodos(root.getRight());
 }
```

Ejemplos

```
Inserte numeros en una misma linea, separados por un espacio: 10 2000 1 44 50 13 205 1231

Listados en preorden:
10
1
2000
44
13
50
205
1231

Listado ordenado:
1
10
13
44
50
205
1231
2000

Cantidad de hojas del arbol: 3

Elemento raiz del arbol: 10

Inserte un numero para encontrar la cantidad de elementos menores estrictos en el arbol: 10

Nodos menores que 10: 1
```

```
Inserte numeros en una misma linea, separados por un espacio: 10 2000 1 44 50 13 205 1231

Listados en preorden:
10
1
2000
44
13
50
205
1231

Listado ordenado:
1
10
13
44
50
205
1231
2000

Cantidad de hojas del arbol: 3

Elemento raiz del arbol: 10

Inserte un numero para encontrar la cantidad de elementos menores estrictos en el arbol: 55

Nodos menores que 55: 5
```