UNIDAD TEMÁTICA 4

Arboles Binarios I

TRABAJO DOMICILIARIO 2

Desarrolla los siguientes algoritmos (método de árbol y método de nodo):

1. Obtener el nodo con la menor clave del árbol.
   - claveMenor(): devuelve un nodo de ABB

```
class TElementoAB<T> implements IElementoAB<T> {
    /**
     * Devuelve el menor elemento.
     *
     * @return TElementoAB<T>.
     */
    @Override
    public TElementoAB<T> obtenerMenorElemento() {
        if (this.getHijoIzq() == null) {
            return this;
        }
        return this.getHijoIzq().obtenerMenorElemento();
    }
```

2. Obtener el nodo con la mayor clave del árbol

- claveMayor(): devuelve un nodo de ABB

```
class TElementoAB<T> implements IElementoAB<T> {
    /**
     * Devuelve el mayor elemento.
     *
     * @return TElementoAB<T>.
     */
    @Override
    public TElementoAB<T> obtenerMayorElemento() {
        // Si es el último a la derecha, es el mayor
        if (this.getHijoDer() == null) {
            return this;
        }
        return this.getHijoDer().obtenerMayorElemento();
    }
```

3. Listar todas las hojas, cada una con su nivel. Usar dos parámetros en el método de nodo: un entero para ir llevando el nivel y una lista Strings “nodo.etiqueta – nivel” para ir agregando las etiquetas de las hojas y su nivel)

- listaDeHojas(): devuelve una lista de String “etiqueta – nivel”

4. Verificar si el árbol es de búsqueda
   - esABB(): devuelve VERDADERO si es de búsqueda, FALSO en caso contrario

```java
public class TArbolBB<T> implements IArbolBB<T> {
    /**
     * Retorna si este es un árbol de búsqueda.
     * @return boolean.*/
    @Override
    public boolean esABB() {
        if (esVacio()) {
            return true;
        } else {
            return raiz.esABB();
        }
    }
```

```java
class TElementoAB<T> implements IElementoAB<T> {
    /**
     * Retorna si este es un árbol de búsqueda.
     * @return boolean.*/
    @Override
    public boolean esABB() {
        if (hijoIzq == null && hijoDer == null) {
            return true;
        } else if (hijoIzq == null && hijoDer != null) {
            return (this.getEtiqueta().compareTo(hijoDer.getEtiqueta()) <
0) == hijoDer.esABB();
        } else if (hijoIzq != null && hijoDer == null) {
            return (this.getEtiqueta().compareTo(hijoIzq.getEtiqueta()) >
0) == hijoIzq.esABB();
        } else {
            return ((this.getEtiqueta().compareTo(hijoDer.getEtiqueta()) <
0) == hijoDer
                    .esABB()) ==
((this.getEtiqueta().compareTo(hijoIzq.getEtiqueta()) > 0) ==
hijoIzq.esABB());
        }
    }
```

5. Obtener la clave inmediata anterior a una clave dada (pasada por parámetro)
   - anterior(Comparable etiqueta): devuelve un nodo del ABB, nulo si la etiqueta del parámetro es la menor del árbol

```
public class TArbolBB<T> implements IArbolBB<T> {
    /**
     * Retorna clave inmediata anterior
     * @param etiqueta clave del elemento
     * @return clave del elemento*/
    @Override
    public Comparable obtenerClaveInmediataAnterior(Comparable
etiqueta) {
        return raiz.obtenerClaveInmediataAnterior(etiqueta);
    }

    /**
     * Retorna clave inmediata anterior
     * @param etiqueta clave del elemento
     * @param predecesor auxiliar para guardar predecesor
     * @return clave del elemento*/
    @Override
    public Comparable obtenerClaveInmediataAnterior(Comparable
etiqueta, Comparable predecesor) {
        return raiz.obtenerClaveInmediataAnterior(etiqueta,
predecesor);
    }
```

```
class TElementoAB<T> implements IElementoAB<T> {
    /**
     * Retorna clave inmediata anterior
     * @param etiqueta clave del elemento
     * @return clave del elemento*/
    @Override
    public Comparable obtenerClaveInmediataAnterior(Comparable
etiqueta) {
        return obtenerClaveInmediataAnterior(etiqueta, null);
    }
```

```java
    /**
     * Retorna clave inmediata anterior
     *
     * @param etiqueta   clave del elemento
     * @param predecesor auxiliar para guardar predecesor
     * @return clave del elemento
     */
    @Override
    public Comparable obtenerClaveInmediataAnterior(Comparable
etiqueta, Comparable predecesor) {
        if (this.etiqueta.equals(etiqueta)) {
            if (this.hijoIzq != null) {
                return
this.hijoIzq.obtenerMayorElemento().getEtiqueta();
            }
        }
        else if (etiqueta.compareTo(this.etiqueta) < 0) {
            if (this.hijoIzq != null) {
                return
this.hijoIzq.obtenerClaveInmediataAnterior(etiqueta, predecesor);
            }
        }
        else {
            predecesor = this.etiqueta;
            if (this.hijoDer != null) {
                return
this.hijoDer.obtenerClaveInmediataAnterior(etiqueta, predecesor);
            }
        }
        return predecesor;
    }
```