Práctica Evaluable - Enero 2021

Estructuras de Datos

Computadores A / Informática D / Matemáticas + Informática

Árboles binarios en Java

Los árboles binarios pueden representarse en Java mediante la siguiente clase `BinTree`:

```
public class BinTree<T extends Comparable<? super T>> {

    private static class Tree<E> {
        private E elem;
        private Tree<E> left;
        private Tree<E> right;

        public Tree(E e, Tree<E> l, Tree<E> r) {
            elem = e;
            left = l;
            right = r;
        }
    }

    private Tree<T> root;
}
```

La clase `BinTree` dispone de hasta 3 constructores y los métodos `isEmpty`, `toString` y `toDot`.

Observa que, aunque el tipo base `T` de los árboles debe ser `Comparable`, los árboles binarios que utilizaremos **no son necesariamente árboles de búsqueda**.

La siguiente figura muestra un árbol binario:

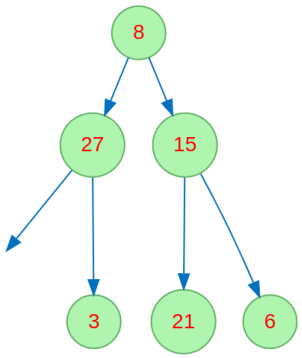

Figura 1: Árbol binario.

Código suministrado

Crea un proyecto Java nuevo en IntelliJ IDEA y descarga el archivo comprimido `Practica-7-Evaluable.src.zip`. Descomprime el archivo y copia las siguientes carpetas en el proyecto Java:

- `src` contiene la platilla `BinTree.java`, la excepción `BinTreeException.java` y el programa de prueba `BinTreeDemo.java`. Solo tienes que modificar y entregar `BinTree.java`
- `dot` una carpeta vacía en la que se almacenarán los árboles generados por `BinTreeDemo`. Solo tienes que pinchar en los ficheros `.dot` para visualizarlos.
- `expectedOutput` contiene la salida que debería aparecer en `dot`. No modifiques su contenido.

Ejercicios

Completa en la clase `BinTree` los siguientes métodos. Cada ejercicio vale 2 puntos.

Ejercicio 1. `T maximum()` que devuelve el máximo valor almacenado en un árbol binario. Para el árbol de la figura 1 el máximo es 27. Si el árbol está vacío debe elevar la excepción `BinTreeException`.

Ejercicio 2. `int numBranches()` que devuelve el número de ramas que forman un árbol (recuerda que una rama es un camino que va desde la raíz a una hoja). Para el árbol de la figura 1 el número de ramas es 3.

Ejercicio 3. `List<T> atLevel(int i)` que devuelve una lista con los elementos del árbol que aparecen en el i-ésimo nivel. Los elementos deben aparecer ordenados en la lista de izquierda a derecha. Si el árbol no tiene elementos en el nivel i-ésimo se devuelve una lista vacía. Para el árbol de la figura 1 se deben obtener las siguientes listas:

```
atLevel(0) = LinkedList(8)
atLevel(1) = LinkedList(27,15)
atLevel(2) = LinkedList(3,21,6)
atLevel(3) = Linked List()
```

Ejercicio 4. Una operación interesante de los árboles binarios es la rotación de los nodos. Por ejemplo, en la siguiente figura se puede apreciar el efecto de **rotar a la izquierda** el nodo `x`. Como puede observarse, el nodo `x` desciende por la izquierda, su hijo izquierdo `y` asciende hasta la raíz y el hijo derecho del nodo `y`, `B` pasa a ser el hijo derecho del nodo `x`.

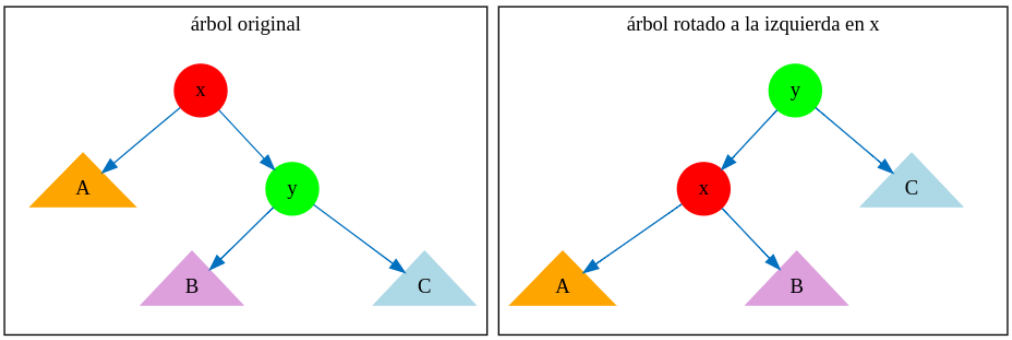

Figura 2: Rotación a la izquierda en x.

La anterior figura muestra la rotación izquierda aplicada al nodo raíz, pero la rotación izquierda puede aplicarse a cualquier nodo del árbol binario, **siempre que su hijo derecho no sea vacío**. La siguiente figura muestra el árbol de la figura 1 con una rotación a la izquierda aplicada en el nodo 8:

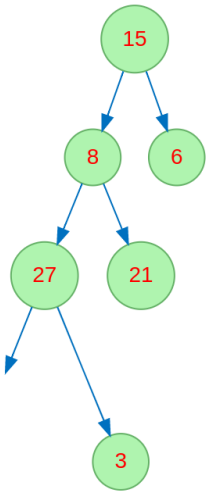

Figura 3: Árbol binario rotado a la izquierda en 8.

Completa la definición del método `void rotateLeftAt(T x)` que rota a la izquierda el árbol en el nodo con valor `x`. Para localizar el nodo `x`, **supondremos que el árbol binario es de búsqueda**. No te preocupes por los árboles que no son de búsqueda; los test los rotan en la raíz. Si la rotación no puede llevarse a cabo porque `x` no aparece en el árbol o porque el hijo derecho de `x` está vacío, el árbol no se modifica.

Ejercicio 5. `void decorate(T x)` que orla el árbol añadiendo todas las posibles nuevas hojas con el valor `x`. Por ejemplo, para el árbol de la figura 1 y el valor 2 debe obtenerse el árbol:

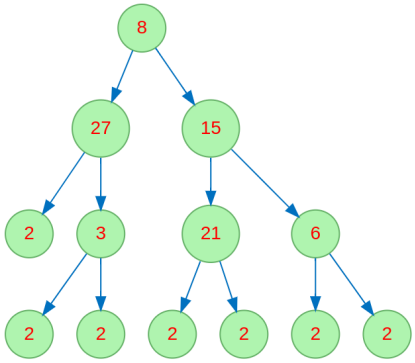

Figura 4: Árbol binario decorado con 2.

Entrega de la práctica evaluable

- Se debe entregar el fichero `BinTree.java`.
- No olvides completar tu nombre, apellidos y grupo.
- Para que un método puntúe es necesario que compile. Si un método no compila, anula tu código con un comentario y déjalo como estaba en la plantilla.