Valor más grande en cada nivel del árbol

El método **largestValueInEachLevel** debe imprimir el mayor valor presente en cada nivel de un árbol binario cuyos nodos contienen números enteros. Ejemplos:

Árbol que invoca el método:

Salida en pantalla: 1 3

Árbol que invoca el método:

Salida en pantalla: 4 9 7

Solución:

Función de ayuda o subfunción

```
//Esta funcion guarda el contenido de todas las hojas de un nivel especifico; en una
//cola de prioridad
public void leavesContentLevel (int i, int level ,int limite, PriorityQueue<T> prioridad){
    //Este if verifica que el arbol que se va a usar no este vacio y que el nivel ingresado
    //no sea mayor al nivel general del arbol
    if(!this.isEmpty()&& level<=limite){
        if(i<level){
            if(this.getRoot().getLeft()!=null){
               this.getRoot().getLeft().leavesContentLevel(i+1, level, limite, prioridad);
            }
            if(this.getRoot().getRight()!=null){
               this.getRoot().getRight().leavesContentLevel(i+1, level, limite, prioridad);
            }
        }
        else if(i==level){
            prioridad.add(this.getRoot().getContent());
        }
    }
}
```

Recursiva:

```
public void largestValueOfEachLevelRecursive(int level){
    //Al llamar al metodo siempre se le debe poner 1 como parametro
    //Ya que es el nivel base de todos los arboles no vacios
    //El metodo se llamara asi mismo hasta que haya llegado al ultimo nivel del arbol
    if(level<=this.countLevelsRecursive()){
        PriorityQueue<T> cola = new PriorityQueue<>();
        this.leavesContentLevel(1, level, this.countLevelsRecursive(), cola);
        System.out.println(cola.toArray()[cola.size()-1]);
        largestValueOfEachLevelRecursive(level+1);
    }
}
```

Iterativa:

```
public void largestValueOfEachLevelIterative(){
    int limite= this.countLevelsRecursive();
    for (int i = 1; i <=limite; i++) {
        PriorityQueue<T> cola = new PriorityQueue<>();
        this.leavesContentLevel(1, i, limite, cola);
        System.out.println(cola.toArray()[cola.size()-1]);
    }
}
```

El método **countNodesWithOnlyChild** debe retornar el número de nodos de un árbol que tienen un solo hijo. Ejemplo:

Árbol que invoca el método:

Valor retornado: 3 (los nodos Seven, Five y Nine tienen un solo hijo).

Solución:

Recursiva:

```
public int recursiveCountNodesWithOnlyChild(){
    if(this.isLeaf()){
        return 0;
    }
    if(this.getLeft()== null) {
        return 1 + this.getRight().recursiveCountNodesWithOnlyChild();
    }
    else if (this.getRight() == null) {
        return 1 + this.getLeft().recursiveCountNodesWithOnlyChild();
    }
    else {
        return root.getLeft().recursiveCountNodesWithOnlyChild() + root.getRight().recursiveCountNodesWithOnlyChild();
    }
}
```

Iterativa:

```
public int iterativeCountNodesWithOnlyChild (){
    int contador_child = 0;
    if (this.isEmpty()) {
        return 0;
    }
    else {
        Stack<BinaryTree<T>> pila = new Stack();
        pila.push(this);
        while (!pila.empty()) {
            BinaryTree<T> arbol_h = pila.pop();
            if (arbol_h.root.getLeft() != null) {
                pila.push(arbol_h.root.getLeft());
            }
            if (arbol_h.root.getRight() != null) {
                pila.push(arbol_h.root.getRight());
            }
            if ((arbol_h.getLeft() !=null || arbol_h.getRight()!=null) && !(arbol_h.getLeft() !=null && arbol_h.getRight()!=null)){
                contador_child++;
            }
        }
    }
    return contador_child;
}
```