

## Teoría

1.-

Vértice	Desde	Coste
0	0	0
1	0	3
2	0	5
3	0	3
4	2	12
5	1	5
6	1	6
7	3	10

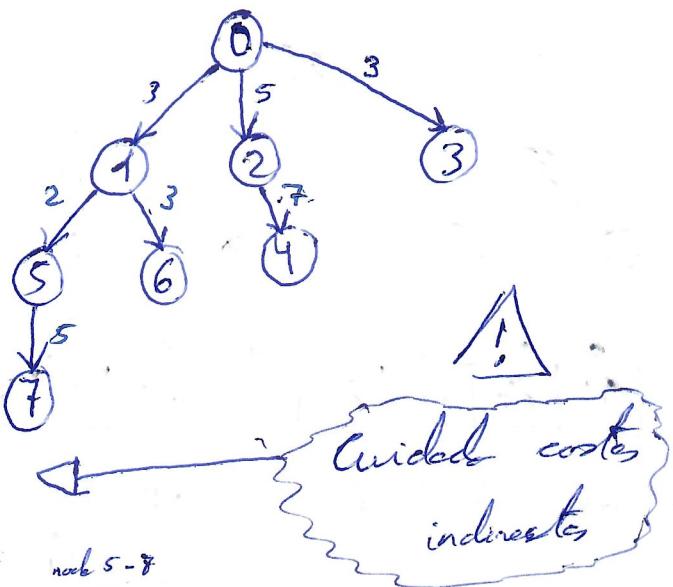
Coste directo

$5 = 3 + 2$

$6 = 3 + 3$

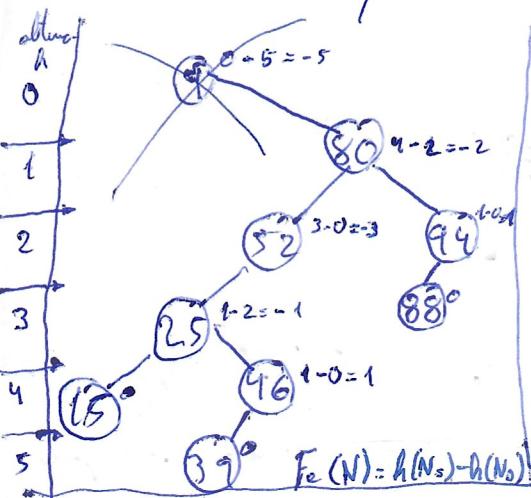
$10 = 3 + 2 + 5$

Coste indirecto

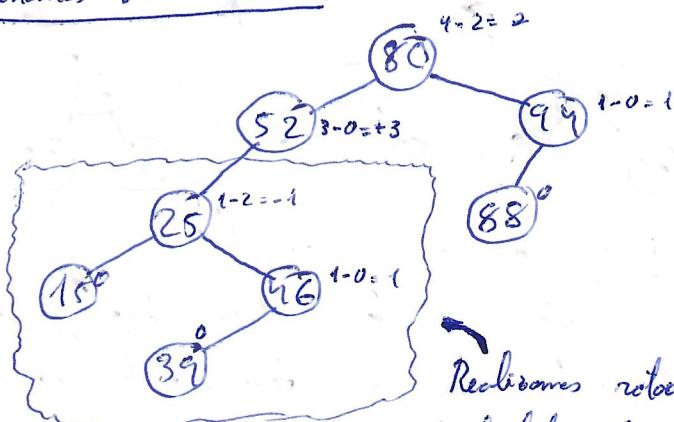


## 2.- ABB (AVL)

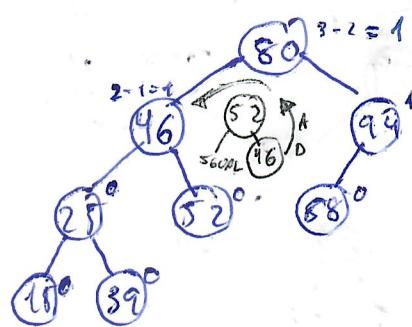
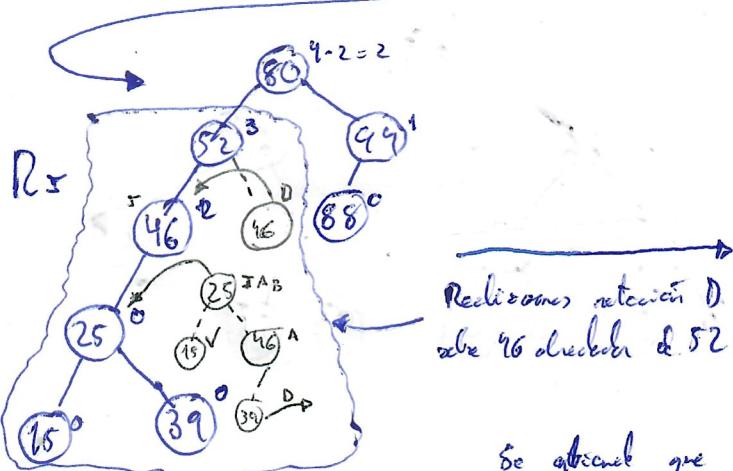
¿Qué elementos hay que eliminar para que la restitución de la condición de equilibrio se logre tras una rotación de tipo 5D?



Eliminamos el nodo 9.



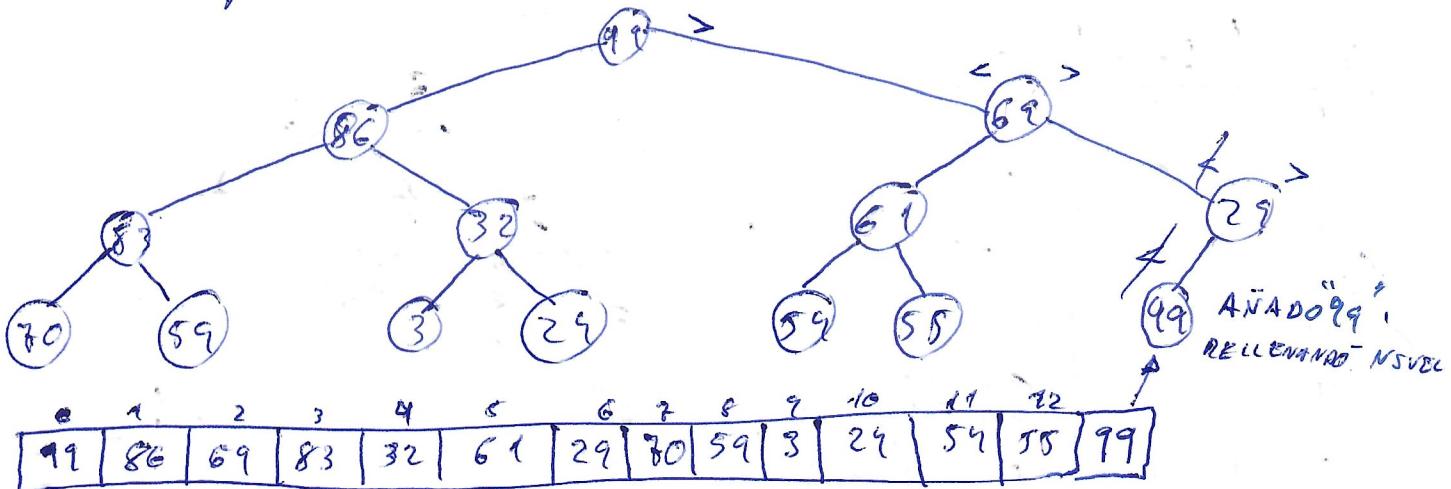
Realizamos rotación 5 sobre 46 alrededor de 25



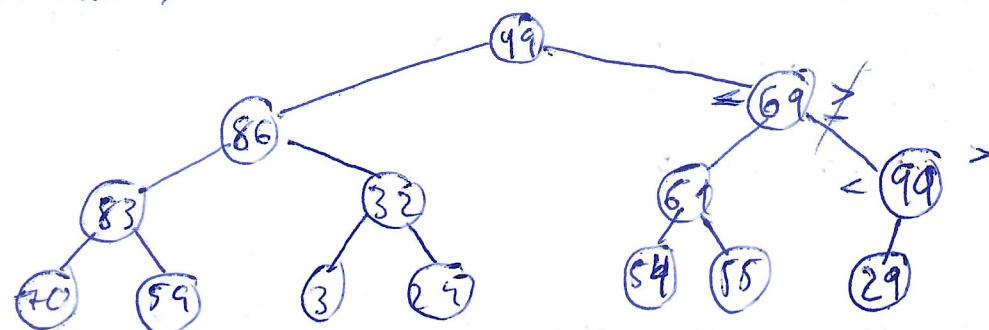
Realizamos rotación D sobre 46 alrededor de 52

Se obtiene que d en derecha 1 sea 1 nodo, se posiciona encima.

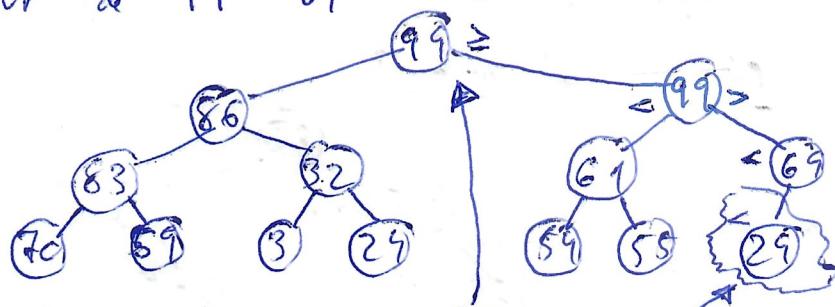
3.- Heap de máximos. ¿Cómo queda tras insertar el elemento 99? A continuación realizan una operación extraer (1) eliminando del árbol el elemento de máxima prioridad.



Una vez añadido, buscamos SPTUP de  $99 \leftrightarrow 29$

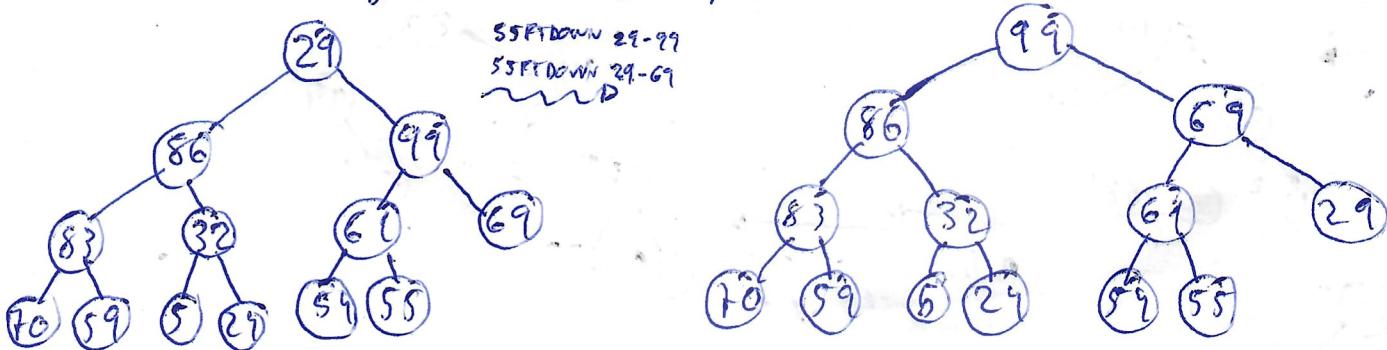


y SPTVP de  $99 \leftrightarrow 69$



Operación extraer máxima prioridad

Se extrae el nodo de la última hoja y se inserta en el nodo raiz.  
El nodo del nodo raiz se descondensan con siftdown, intercambiando nodo padre por uno de sus hijos hasta llegar propiedad de heap



4.

0
1
2
3
4
5
6 11
7
8
9
10 14
11

función hash  $H_1(x) = x \% 11$

insersion 3, 14, 25

función secundaria  $H_2(x) = 7 - x \% 7$

$$H_1(3) = 3 \% 11 = 3$$

$$H_2(3) = 7 - 3 \% 7 = 7 - 3 = 4$$

$$\begin{aligned} \text{Intento } i=0 \rightarrow H(3,0) &= (H_1(3) + 0 \cdot H_2(3)) \% 11 \\ &= (3 + 0) \% 11 = \\ &= 3 \end{aligned}$$

$$H_1(14) = 14 \% 11 = 3$$

$$H_2(14) = 7 - 14 \% 7 = 7 - 0 = 7$$

$$\begin{aligned} \text{Intento } i=0 \rightarrow H(14,0) &= (H_1(14) + 0 \cdot H_2(14)) \% 11 = \\ &= 3 + 0 \% 11 = \\ &= 3 \rightarrow \text{Posición 3 ocupada} \end{aligned}$$

$$\begin{aligned} \text{Intento } i=1 &= H(14,1) = (H_1(14) + 1 \cdot H_2(14)) \% 11 = \\ &= 3 + 7 \% 11 \end{aligned}$$

$$\frac{25}{22} \overbrace{\quad}^{11} \frac{2}{\cancel{3}} \quad = 10$$

$$H_1(25) = \underline{(25 \% 11)} = 3$$

$$H_2(25) = 7 - 25 \% 7 = 3$$

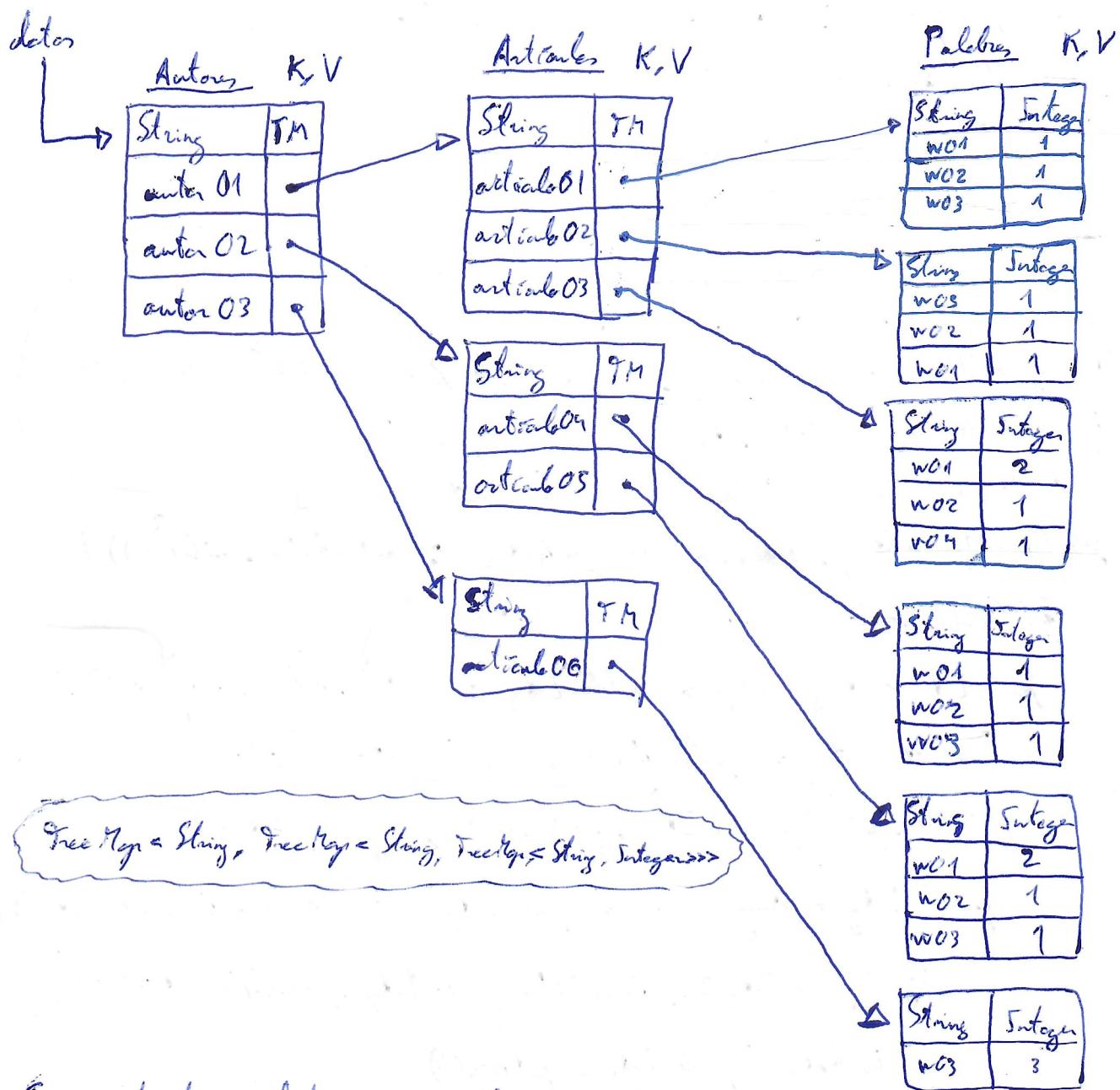
$$\begin{aligned} \text{Intento } i=0 \rightarrow H(25,0) &= ((H_1(25) + 0 \cdot H_2(25)) \% 11 \\ &= (3 + 0) \% 11 = \\ &= 3 \rightarrow \text{Posición 3 ocupada} \end{aligned}$$

$$\begin{aligned} \text{Intento } i=1 \rightarrow H(25,1) &= ((H_1(25) + 1 \cdot H_2(25)) \% 11 \\ &= (3 + 3) \% 11 \\ &= 6 \end{aligned}$$

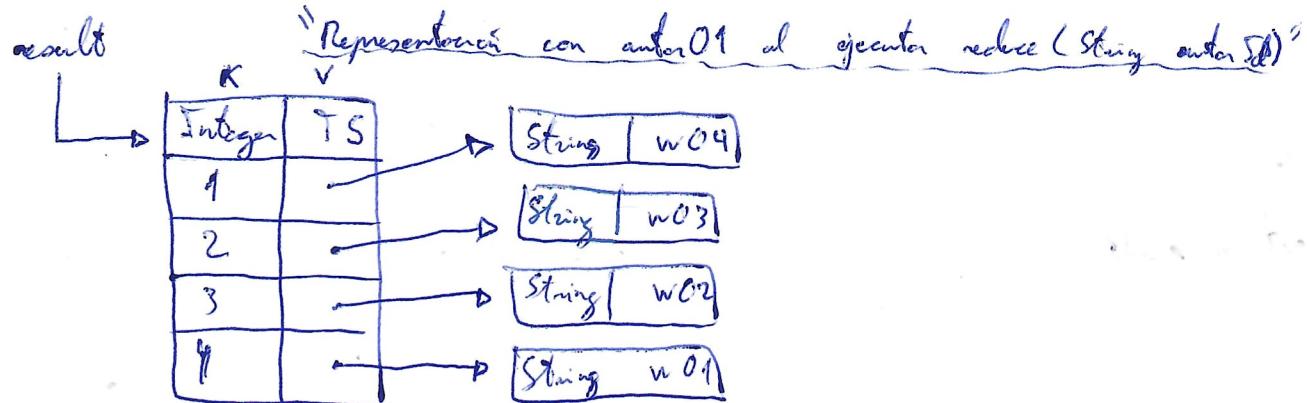
# Práctica

1.-

Se nos pide realizar el método reduce() a partir de las siguientes estructuras.



Cujo resultado sera la siguiente:



```

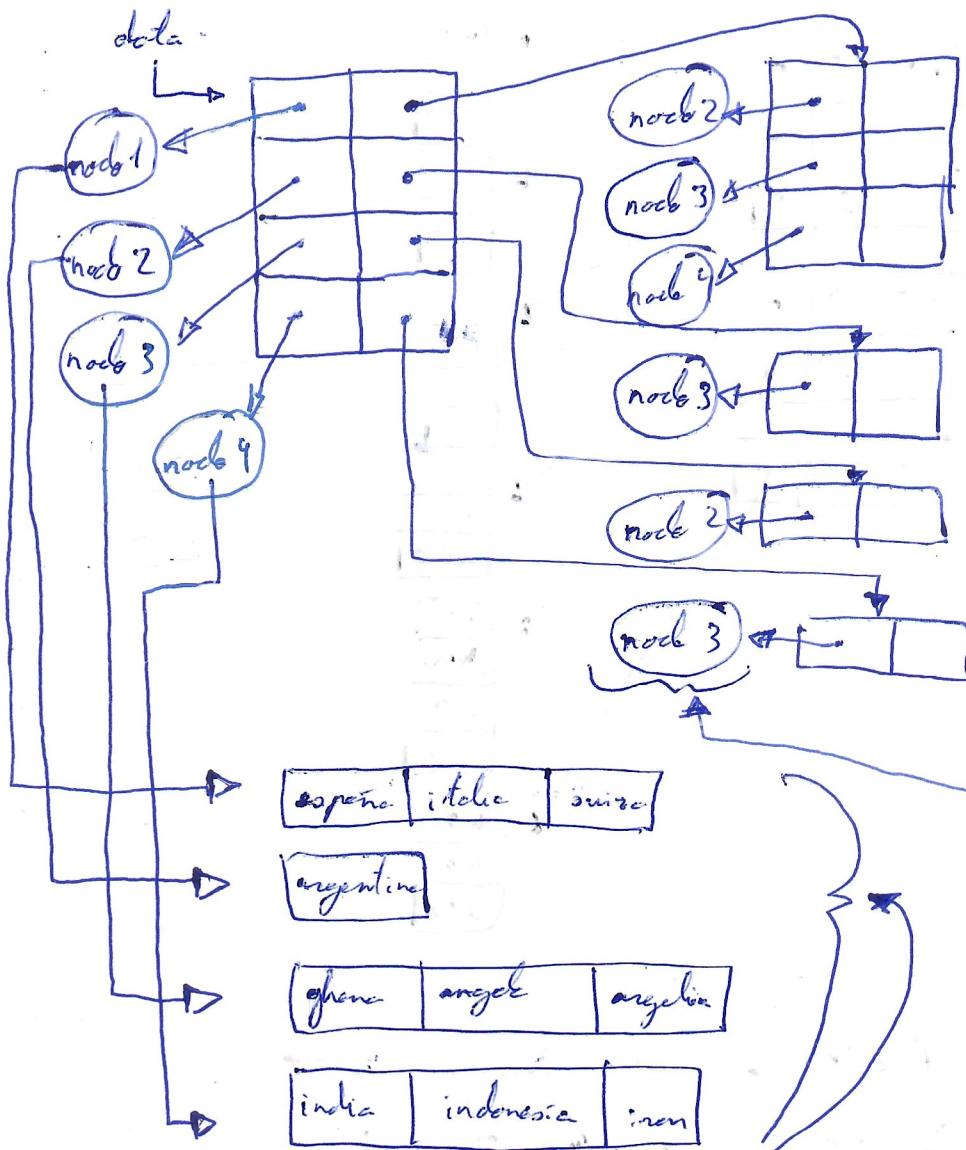
private TreeMap<String, TreeMap<String, TreeMap<String, Integer>> datos = new TreeMap<>();
public void add... {}

public TreeMap<String, TreeSet<String>> reduce (String autor) {
    // Prohibido containsKey()
    // Permitida V <- get(K) y Void <- put(K, V)
    // Prohibido KeySet()
    // Permitida entrySet() y values()
    // Socemos autor actual con sus artículos y palabras
    TreeMap<String, TreeMap<String, Integer>> autorActual = datos.get(autor);
    if (autorActual == null) return new TreeMap<>();
    // Recorremos los artículos de ese autor
    for (TreeMap<String, Integer> articulos : autorActual.values()) {
        // Recorremos las palabras de ese artículo
        for (Entry<String, Integer> palabraEntry : articulos.entrySet()) {
            TreeSet<String> palabraActual = new TreeSet<>();
            resultado.put(palabraEntry.getKey(), palabraActual);
            for (Entry<String, Integer> palabraEntry2 : articulos.entrySet()) {
                if (palabraEntry2.getValue() != palabraEntry.getValue()) continue;
                palabraActual.add(palabraEntry2.getKey());
            }
        }
    }
    return resultado;
}

```

2.

1) Representación gráfica de la estructura de datos  $\text{Tree}_{\text{Ty}} \prec \text{Node} \prec \text{Tree}_{\text{Ty}}$ ,  $\text{Tree}_{\text{Ty}} \prec \text{Node} \prec \text{Tree}_{\text{Ty}}$ ,  $\text{String} \gg$



ESTA REPRESENTACIÓN DEBERÍA SER AL REVÉS.  
!

Se entiende que estos Strings también están en los nodos que se llaman igual. Es decir, para node2 y cualquier node 2, tiene como String argentina.

2) Tomando como partida la representación gráfica y su entendimiento, la salida por consola debería ser:

$$[\text{argentina}] \rightarrow [\text{ghana}, \text{angola}, \text{angelia}] = 1$$

$$[\text{espana}, \text{italia}, \text{suiza}] \rightarrow [\text{argentina}] = 3, [\text{ghana}, \text{angola}, \text{angelia}] = 1, [\text{india}, \text{indonesia}, \text{iran}] = 1$$

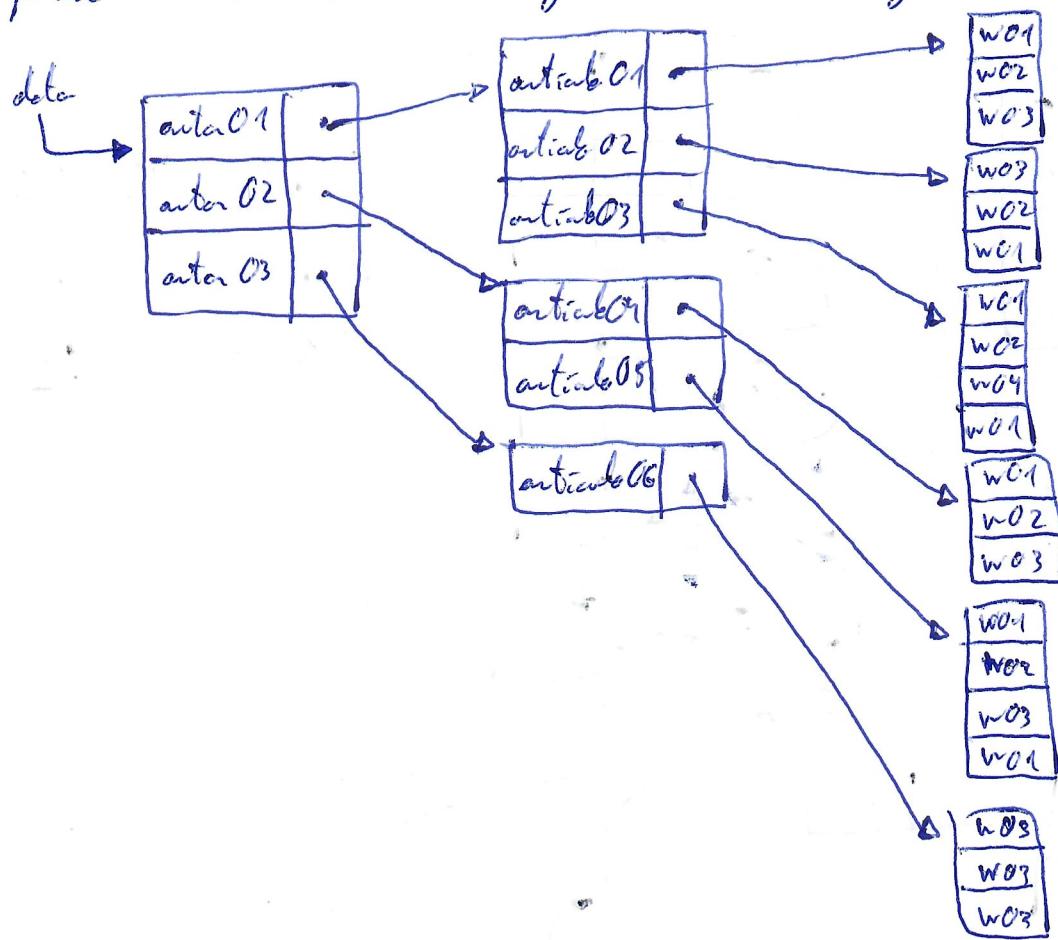
$$[\text{ghana}, \text{angola}, \text{angelia}] \rightarrow [\text{argentina}] = 2$$

$$[\text{india}, \text{indonesia}, \text{iran}] \rightarrow [\text{ghana}, \text{angola}, \text{angelia}] = 1$$

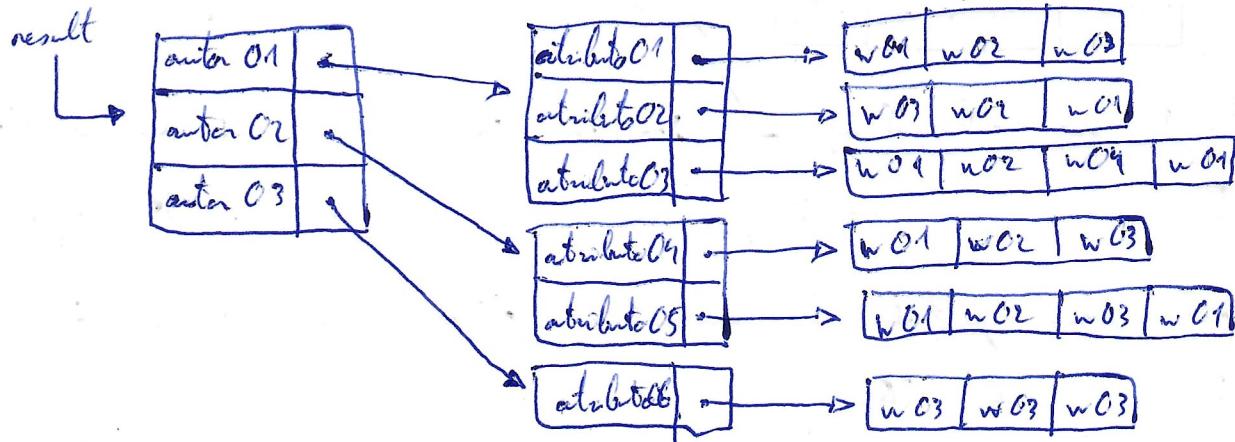
El contexto es que tenemos node2, node3 y node4 con sus correspondientes nodos 1, 2, 3 y 4

3.-

Se nos pide crear el método convertir() con la siguiente estructura de partida  $\rightarrow$  AVLTree < Par < String, AVLTree < Par < String, AVLTree < Par < String  $\gg\gg\gg$  dato.



El método convertir transformará esta estructura en  
arraylist < Par < String, arraylist < Par < String, arraylist < String  $\gg\gg\gg$  result



```
public arraylist < Par < String, arraylist < Par < String, arraylist < String>>>> convertir() {
    arraylist < Par < String, arraylist < Par < String, arraylist < String>>>> result = new arraylist <> ();
    for (Par < String, AVLTree < Par < String, AVLTree < String>>>> autor : dato) {
        arraylist < Par < String> arraylist < String>>> atributoResult = new arraylist <> ();
        for (Par < String, AVLTree < String>>> atributo : autor) {
            arraylist < String> paraleloResult = new arraylist <> (atributo.getValue());
            atributoResult.add(new Par <> (atributo.getKey(), atributoResult));
        }
        result.add(autor.getKey(), atributoResult);
    }
    return result;
}
```