Resolva cada exercício em folhas separadas

4 p^{tos} **1.** Considere uma lista de objetos Pair<Pais,Pais> em que cada par representa uma fronteira entre dois países. Pretende-se que contabilizem, **da forma mais eficiente**, o número de fronteiras de cada país devolvendo num map ordenado por ordem crescente do número de fronteiras. Por exemplo, para a lista de fronteiras lf={<p1, p2>, <p2, p3>, <p3, p4>, <p5, p3>, <p5, p6>} deve devolver 1 -> p1, p4, p6; 2 -> p2, p5; e 3 -> p3.

```
Map<Integer,ArrayList<Pais>> numBorders (List<Pair<Pais,Pais>> lf)
```

3 p^{tos} **2.** Seja o seguinte método:

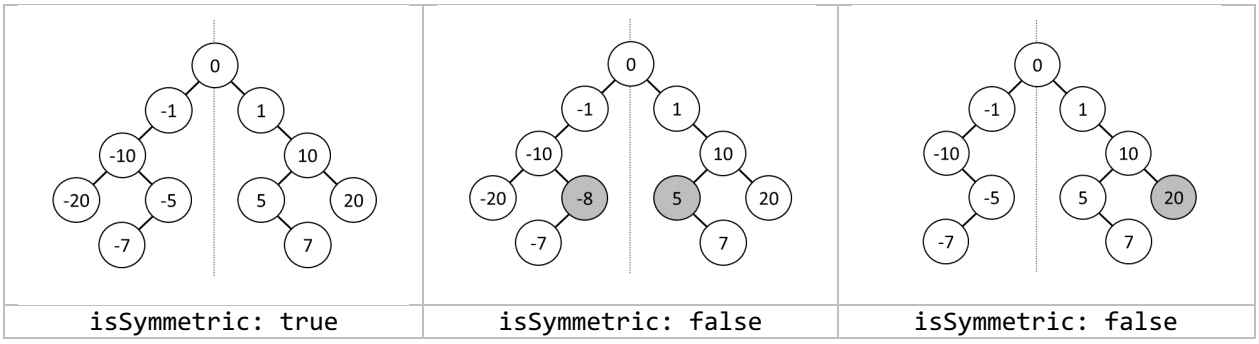
```
static public void mistery (int n) {
    if (n > 0) {
        for (int i=1; i<=n; i=i*2) {
            processa(i); // método de complexidade O(1)
        }
        mistery(n-1);
    }
}
```

Analise o método quanto à sua complexidade temporal, utilizando a notação Big-Oh. Justifique adequadamente.

5 p^{tos}

3. Implemente o método `boolean isSymmetric()` na classe `MyTree` que verifica se a árvore é simétrica. O método **deve validar não só a topologia da árvore**, mas também **os valores nela contidos**. A comparação entre os valores deve ser efetuada por valor absoluto, isto é, à esquerda do nó raiz apenas há valores negativos e à direita deve corresponder o respetivo valor positivo.

```
public class MyTree extends BST<Integer> {
    public boolean isSymmetric(){
        ...
    }
}
```



isSymmetric: true	isSymmetric: false	isSymmetric: false

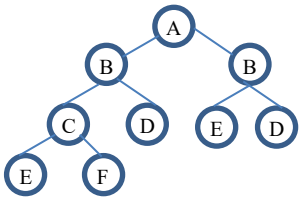
Resolva cada exercício em folhas separadas

5 ptos **4.** Um problema frequente nos trajetos de uma rede ferroviária é não ser possível passar em determinadas estações por estas se encontrarem em manutenção. Considere a classe Ferrovia que implementa uma **rede ferroviária** e as suas várias estações através de um **grafo dirigido** usando a classe map de adjacências com as estações representadas por uma String que as identifica.
Pretende-se um método, **o mais eficiente possível**, que recebe duas estações, origem e destino, uma estação em manutenção e retorna um caminho que não passa pela estação em manutenção.

```
public class Ferrovia {
    private Graph<String,Double> g = new Graph<>(true);

    public LinkedList<String> alternativePath(String stOrig, String stDest,
                                                          String stManut)
}
```

3 ptos **5.** Implemente na classe HeapPriorityQueue<K,V> um método que dados dois índices de valores numa heap, devolve a lista com os elementos comuns que ficam no caminho (comum) desses dois valores até à raiz.



Heap: [A,B,B,C,D,E,D,E,F]
0 1 2 3 4 5 6 7 8
(4, 8) → {B, A}
(5, 8) -> {A}
(7, 8) -> {A,B, C}

```
public  List<Entry<K,V>> getCommonPathElements(int idx1, int idx2)
```