Ejercicio 3

Se tienen n monedas de igual tamaño. Se sospecha que una de ellas es falsa. Para detectarlo, se sabe que todas las monedas deberían tener el mismo peso, y que la falsa tendría un peso diferente (mayor o menor). Considere que dispone de una lista que almacena los objetos de tipo Moneda y de un método, de complejidad constante, que permite hacer una pesada de las monedas:

int pesada (List<Moneda> lm, int i_izq, int j_izq, int i_der, int j_der)

Este método devuelve un número positivo si las monedas en el intervalo *[i_izq, j_izq)* pesan más que las del intervalo *[i_der, j_der)*, cero si pesan igual ambos platos, y un número negativo en otro caso. Diseñar una función recursiva que determine, con el mínimo número de pesadas posible, el índice en la lista de la moneda falsa si existe, o -1 si no hay ninguna falsa, así como si la moneda falsa es más o menos pesada (valor negativo si es menos pesada, positivo si es más pesada, o 0 si no hay ninguna falsa). La función tendrá **orden logarítmico** y la siguiente signatura:

Tuple2<Integer,Integer> monedaFalsa(List<Moneda> lm)

***Ejemplo**: Para la lista de monedas [7,7,5,7,7,7,7,7,7], la solución sería (2,-2).*

Notas:

- Se aconseja formar 3 grupos con el mismo número de monedas
- n debe ser mayor que 2 y potencia de 3
- supóngase implementado el método necesario para determinar la solución en el caso base, con nombre “casoBase”

SE PIDE

a) Dar una definición recursiva de la función solicitada e implementarla en Java.
b) Sin necesidad de codificarlo, indique lo más detalladamente posible qué debe realizar el método del caso base.
c) Justifique la complejidad del algoritmo mediante su cálculo.
d) ¿Qué modificaciones serían necesarias si n no fuera potencia de 3?

SOLUCIÓN

Apartado a)

$$monedaFalsa(mn) = \{mf(mn, 0, mn.size(\ \ ))$$

$$mf(mn,i,j) = \begin{cases} casoBase(mn,i,j), & k = (j-i)/3 \\ mf(mn,i+2*k,j), & j-i \leq 3 \\ mf(mn,i+k,i+2*k), & pesada(mn,i,i+k,i+k,i+2*k) == 0 \\ mf(mn,i,i+k), & pesada(mn,i,i+k,i+2*k,i+3*k) == 0 \\ & e.o.c. \end{cases}$$

```
public static Tuple2<Integer,Integer> monedaFalsa(List<Integer> mn) {
	return mf(mn,0,mn.size());
}

public static Tuple2<Integer,Integer> mf(List<Integer> mn, Integer i, Integer j) {
	Tuple2<Integer, Integer> res = null;
	if (j-i <= 3) {
		res = casoBase(mn,i,j);
	} else {
		int k = (j-i)/3;
		int p1 = pesada(mn,i,i+k,i+k,i+2*k);
		int p2 = pesada(mn,i,i+k,i+2*k,i+3*k);
		if (p1==0) {
			res = mf(mn,i+2*k,j);
		} else if (p2==0) {
			res = mf(mn,i+k,i+2*k);
		} else {
			res = mf(mn,i,i+k);
		}
	}
	return res;
}
```

Apartado b)

Dadas las monedas [a,b,c]
Si todas las monedas son iguales, devolver -1.
En caso contrario, es necesario pesar dos partes:
- Si a y b son iguales, la falsa es c y se hace otra pesada para comprobar si c pesa más o menos.
- Si a y b no son iguales, se coge la de más peso de las dos y se realiza una pesada con la parte c.
  - Si son iguales, la falsa es la que pesa menos de a y b.
  - Si no son iguales, la falsa es la más pesa de la primera pesada y siendo su peso el mayor.

Apartado c)

Tamaño del problema n = j – i
$T(n) = T(n\ /\ 3) + k$
$T(n) \in \Theta(log)$

Apartado d)

Si el código contempla que al caso base siempre se va a llegar con 3 elementos, no hay que cambiar nada.
Si el código no contempla que al caso base siempre se va a llegar con tres elementos:
- Si el grupo que se tiene es mayor o igual a 3 elementos, habría que llamar al caso base.
- Si el grupo que se tiene es menor a 3 elementos, sería necesario incluir y 1 ó 2 monedas más para formar el grupo de 3 elementos y llamar al caso base.