





**30 de junio de 2021** 

N° matrícula:	Nombre:	
Apellidos:		

**Problema**. Sean A y B dos vectores de N elementos enteros, ordenados circularmente y que pueden contener números repetidos. Ambos vectores comparten exactamente los mismos elementos hasta una posición determinada 'k', **a partir de la cual todos sus elementos serán diferentes**. Se pide implementar un algoritmo, que dado los vectores A y B determine esa posición 'k'. En el caso de que los dos vectores sean idénticos el procedimiento devolverá -1 (indicando de esa forma que tal posición no existe).

a) Diseñar el procedimiento basado en Divide y Vencerás con complejidad O(log N) en el caso peor¹ (donde N es el tamaño del vector) que devuelva un número entero que corresponde a la posición del primer elemento diferente entre ambos vectores.

## Ejemplo:

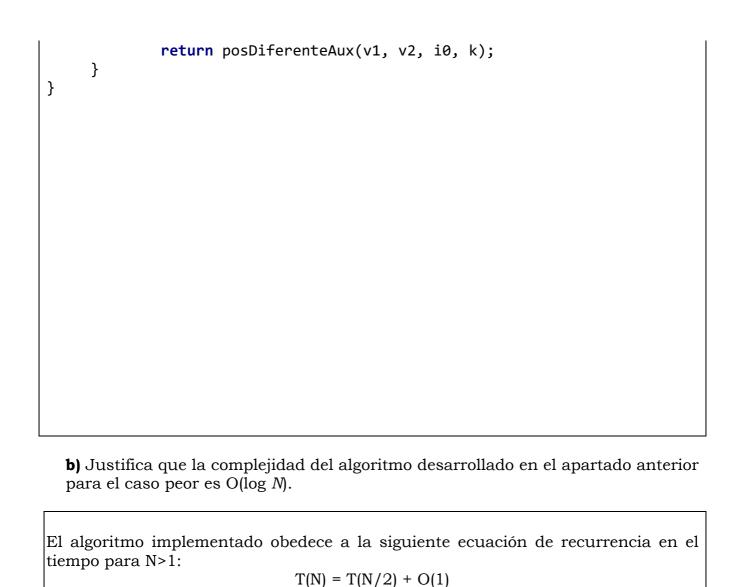
0	I	2	3	4	5	6	7	8
-4	-2	0	I	I	2	3	4	-9
0	1	2	3	4	5	6	7	8
-4	-2	0	I	I	5	7	10	-6

Devolvería la posición=5

```
int posDiferente (int[] vector1, int[] vector2)
```

```
int posDiferente (int[] v1,int[] v2) {
         return posDiferenteAux(v1, v2, 0, v1.length-1);
}
int posDiferenteAux(int v1[], int[] v2, int i0, int iN) {
     boolean encontrado = false;
     if (i0==iN) { //0(1)
         if (v1[i0] == v2[i0])
                 return -1;
         else
             return i0;
     } else {
         int k = (i0 + iN) / 2; //0(1)
         //O(T(N/2) N>1, 2^{\circ} caso TM, Log(N)
         if(v1[k] == v2[k]) //busco lado derecho
             return posDiferenteAux(v1, v2, k+1, iN);
         else //busco lado izquierdo;
```

<sup>&</sup>lt;sup>1</sup> Desarrollar un algoritmo que tenga una complejidad diferente a O(log N) en el caso peor conllevará una puntuación de 0 en la pregunta.



Esta ecuación es del tipo  $T(N) = p \cdot T(N/q) + f(N)$ , donde  $f(N) \in O(N^a)$ , con p=1, q=2 y a=0, por lo que podemos aplicar el Teorema Maestro. Dado que  $log_q(p) = log_2(1)=0$  y a=0 nos encontramos en el caso  $2^\circ$  del Teorema maestro ( $a=log_q(p)$ ), por lo que la

complejidad del algoritmo es:  $T(N) \in O(N^{\log_q(p)} \cdot \log N) = O(N^0 \log N) = O(\log N)$