Examen de Algorítmica y Complejidad (Plan 2014) — **17 de octubre de 2022**

Nº matrícula:_____ **Nombre:** _____

Apellidos: _____

Problema. Decimos que un array, *v*, de N enteros está polarizado si todos sus elementos son distintos de 0, todos los elementos negativos ocupan posiciones consecutivas bajas del vector y todos los elementos positivos ocupan posiciones consecutivas altas del vector.

Ejemplo de vector polarizado:

0	1	2	3	4	5	6	7	8
-4	**-7**	**-2**	**34**	**5**	**8**	**7**	**1**	**13**

a) Diseñar un algoritmo basado en **Divide y Vencerás** con complejidad en el caso peor[1] de O(log N) (donde N es el tamaño del vector) que, dado un vector polarizado, devuelva la posición más baja que ocupa un elemento positivo (en el ejemplo anterior el método deberá devolver 3). Se deberán contemplar las situaciones en las que el vector sólo contenga números positivos o sólo contenga números negativos (en este último caso el método deberá devolver -1).

```
int primerPositivoEnPolarizado (int[] vector)
```

```
int primerPositivoEnPolarizado(int[] vector){
    if (vector[0]>0) return 0;                          //sólo hay positivos
    else if (vector[vector.length-1]<0) return -1;    // sólo hay negativos
    else // nos aseguramos que el vector tiene números positivos y negativos
       return primerPositivoEnPolarizadoAux(vector, 0, vector.length-1);
}

int primerPositivoEnPolarizadoAux(int[] vector, int i0, int iN){
    if (i0==iN) return i0;
    else{
        int k=(i0+iN)/2;
        if (vector[k] < 0)
            return primerPositivoEnPolarizadoAux(vector, k+1, iN);
        else
            return primerPositivoEnPolarizadoAux(vector, i0, k);
    }
}
```

[1] Desarrollar un algoritmo que tenga una complejidad diferente a O(log N) en el caso peor conllevará una puntuación de 0 en la pregunta.

Otra opción sería:

```
int primerPositivoEnPolarizado1(int[] vector){
    return primerPositivoEnPolarizadoAux1(vector, 0, vector.length-1);
}

int primerPositivoEnPolarizadoAux1(int[] vector, int i0, int iN){
    if (i0==iN) {
        if (vector[i0] > 0) return i0;
        else return -1;
    }else{
        int k=(i0+iN)/2;
        if (vector[k] < 0)
            return primerPositivoEnPolarizadoAux1(vector, k+1, iN);
        else
            return primerPositivoEnPolarizadoAux1(vector, i0, k);
    }
}
```

b) Justifica que la complejidad del algoritmo desarrollado en el apartado anterior para el caso peor es O(log N).

El algoritmo implementado obedece a la siguiente ecuación de recurrencia en el tiempo para N>1:

$$T(N) = T(N/2) + O(1)$$

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 2º del Teorema maestro ($a=\log_q(p)$), por lo que la complejidad del algoritmo es: $T(N) \in O\left(N^{\log_q(p)} \cdot \log N\right) = O(N^0 \log N) = O(\log N)$