Examen de Algorítmica y Complejidad (Plan 2014) **26 de noviembre de 2021**

Nº matrícula:________________ **Nombre:** ____

Apellidos: ____

Problema. Dado un *array* de números enteros, encontrar la longitud del subarray[1] ordenado más largo. Ejemplo:

0	1	2	3	4	5	6	7	8
2	**1**	**1**	**3**	**2**	**4**	**7**	**15**	**1**

En este ejemplo, la longitud del *subarray* ordenado más largo es 4 (*subarray* 2, 4, 7 y 15).

a) Diseñar un algoritmo basado en *Divide y Vencerás*[2] con complejidad O(N·log N) en el caso peor[3] (donde N es el tamaño del *array*) que devuelva la longitud del *subarray* pedido.

```
int longMaxSubArrayOrdenado(int[] vector)
```

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

int longMaxSubArrayOrdenadoCruzada(int[] vector, int i0, int k, int iN){
     int i=k;
     while (i>i0 && vector[i-1]<=vector[i]) {
         i--;
     }
     int j=k;
     while (j<iN && vector[j+1]>=vector[j]) {
         j++;
     }
     return j-i+1;
 }

int longMaxSubArrayOrdenadoAux(int[] vector, int i0, int iN){
     if (i0==iN)
         return 1;
     else {
         int k = (i0 + iN) / 2;
         int m1 = longMaxSubArrayOrdenadoAux(vector, i0, k);
         int m2 = longMaxSubArrayOrdenadoAux(vector, k + 1, iN);
         int m3 = longMaxSubArrayOrdenadoCruzada(vector, i0, k, iN);
         return Math.max(m1, Math.max(m2, m3));
     }
 }
```

[1] Dado v un array de longitud N y w un array de longitud $M \leq N$. Decimos que w es un subarray de v si y solo si $\exists k \in \{0,\dots N\text{-}M\}$ tal que $\forall i \in \{0,\dots M\text{-}1\}$ $v[k+i]=w[i]$.
[2] Desarrollar un algoritmo que no esté basado en la estrategia divide y vencerás conllevará una puntuación de 0 en todo el problema 1.
[3] Desarrollar un algoritmo con una complejidad diferente a O(Nlog N) en el caso peor conllevará una puntuación de 0 en la pregunta.

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

El peor caso sucede cuando el vector está ordenado. Veamos la complejidad del algoritmo en el peor caso:

* La complejidad del algoritmo longMaxSubArrayOrdenadoCruzada es O(N) en el peor caso ya que el número de iteraciones de los dos bucles while coincide con N, la longitud del vector.

* En este caso el algoritmo longMaxSubArrayOrdenadoAux obedece a la siguiente ecuación de recurrencia en el tiempo:

$$T(N)= 2\cdot T(N/2) + O(N) \text{ para } N>1$$

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

* El algoritmo longMaxSubArrayOrdenado tiene la misma complejidad que longMaxSubArrayOrdenadoAux. Por tanto, tiene complejidad O(N·log N).