Examen de Algorítmica y Complejidad (Plan 2014)

30 de junio de 2023

Nº matrícula:_______________ **Nombre:** ____________________________________

Apellidos: ____

Problema. Dada una cadena de ceros y unos, encontrar la secuencia más larga de unos basado en la estrategia de **Divide y Vencerás** con complejidad en el caso peor[1] de **O(N·logN)** (donde N es el tamaño del vector). Se pide implementar un algoritmo que deberá tener la siguiente cabecera:

```
int oneSubArray (int[] v)
```

donde v es el array que contiene la secuencia de ceros y unos.

Ejemplos:

v1	[1, 1, 1, 1, 1, 0, 1]	**SOL = 5**
v2	[0, 0, 0, 0, 1, 1, 1]	**SOL = 3**
v3	[1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0]	**SOL = 5**
v4	[0, 0, 0, 0, 0]	**SOL = 0**
v5	[1, 0, 1, 0, 1, 0, 1, 0, 1]	**SOL = 1**

```
int oneSubArray(int[] v){
    return oneSubArrayAux(v, 0, v.length-1);
}

int oneSubArrayAux(int[] v, int i0, int iN){
    if(i0==iN){
        return v[i0];
    } else{
        int k = (i0 + iN) / 2;
        int l1 = oneSubArrayAux(v, i0, k);
        int l2 = oneSubArrayAux(v, k+1, iN);
        int l3 = oneSubArrayCruzada(v, i0, k, iN);
        return Math.max(l1, Math.max(l2, l3));
    }
}

int oneSubArrayCruzada(int[] v, int i0, int k, int iN) {
    int len_i = 0;
    int len_j = 0;
    int i=k;
    int j=k+1;

    while(i >= i0 && v[i] == 1){
        len_i++;
```

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

```
        i--;
    }
    while(j <= iN && v[j] == 1){
        len_j++;
        j++;
    }
    int full_len = len_i + len_j;
    return full_len;
}
```

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 todos los elementos del vector son 1. Veamos la complejidad del algoritmo en el peor caso:

* La complejidad del algoritmo oneSubArrayCruzada 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 oneSubArrayAux 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 oneSubArray tiene la misma complejidad que oneSubArrayAux. Por tanto, tiene complejidad O(N·log N).