Examen de Algorítmica y Complejidad (Plan 2014)

18 de enero de 2022

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**	**3**	**2**	**2**	**4**	**4**	**15**	**1**

En este ejemplo, la longitud del *subarray* ordenado más largo es 5 (*subarray* 2, 2, 4, 4 y 15).

Diseñar un algoritmo basado en *Programación Dinámica*[2] con complejidad[3] O(*N*) (donde *N* es el tamaño del *array*) que devuelva la longitud del *subarray* pedido.

a) Define la entrada, la salida y la semántica de las funciones sobre las que estará basado el algoritmo de programación dinámica.

Dado el vector v de longitud n, definimos las dos siguientes funciones:

Max_Total(i): Longitud mayor de los subarrays ordenados comprendidos entre las posiciones 0…i.
Max_End(i): Longitud mayor de los subarrays ordenados que empiezan en algún entre 0…i y terminan en i.

Entrada: i {0…n-1}. El valor i=0 hace referencia a la posición 0 del array.
Salida: valor entero en el rango {1…n} (indica la longitud mayor).

b) Expresa recursivamente las funciones anteriores

Si i=0
    Max_End(0)=1
    Max_Total(0)=1
Si i>0
    Max_End(i) = 1 si v[i-1]>v[i]
    Max_End(i) =1+Max_End(i-1) si v[i-1]≤v[i]

    Max_Total(i) = max{Max_Total(i-1), Max_End(i) }

[1] Dado *v* un array de longitud *N* y *w* un array de longitud *M*≤*N*. Decimos que *w* es un subarray de *v* si y solo si $\exists k \in \{0,\ldots N\text{-}M\}$ tal que $\forall i \in \{0,\ldots M\text{-}1\}$ $v[k+i]=w[i]$.

[2] Desarrollar un algoritmo que no esté basado en la estrategia Programación Dinámica conllevará una puntuación de 0 en todo el problema 2.

[3] Desarrollar un algoritmo que no tenga complejidad O(*N*) conllevara una puntuación de 0 en todo el problema 2.

c) Implementa el algoritmo basado en *Programación Dinámica* con complejidad en memoria O(1).

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

```
int longMaxSubArrayOrdenado(int[] vector){
   int max_end=1;
   int max_total=1;
   for (int i=1;i<vector.length; i++)
      if (vector[i-1]>vector[i])  max_end = 1;
      else max_end = max_end + 1;

      max_total=Math.max(max_end, max_total);
   }
   return max_total;
}
```