# Examen de Algorítmica y Complejidad (Plan 2014)

28 de junio de 2019

**Nº matrícula:**_____ **Nombre:** _____

**Apellidos:** _____

**Problema**. Nos dan un *vector* de *N* números enteros, y un número *K*. Podemos transformar el vector multiplicando por -1 uno de sus elementos. Deseamos transformar el vector *K* veces exactas de manera que la suma de sus elementos sea máxima.

Ejemplo:

*K:* 4

*vector*:

| | | | | |
|---|---|---|---|---|
| -2 | 0 | 5 | -1 | 2 |

La solución es:

| | | | | |
|---|---|---|---|---|
| 2 | 0 | 5 | 1 | 2 |

Detalle de la solución:

| | | | | |
|---|---|---|---|---|
| -2 | 0 | 5 | -1 | 2 |
| +2 | 0 | 5 | -1 | 2 |
| 2 | 0 | 5 | +1 | 2 |
| 2 | - 0 | 5 | 1 | 2 |
| **2** | **+0** | **5** | **1** | **2** |

Se pide implementar en Java un algoritmo basado en un **esquema voraz** que **resuelva**[1] el problema (esto es que encuentre el vector de suma máxima tras *K* transformaciones) con complejidad en tiempo[2] O(*N*·*K*) donde *N* es el tamaño del vector y *K* el número de transformaciones pedidas:

```
int[] vectorTransformado(int[] vector, int K)
```

donde:

- **`vector`**: se refiere al vector de la entrada del problema.
- **`K`**: se refiere al número de transformaciones que tenemos que hacer (esto es multiplicar por -1 un elemento del vector).
- La función debe devolver el **vector con suma máxima** tras *K* transformaciones.

a) Explicar la estrategia voraz que permite **resolver** el problema.

Cada vez que se transforme el vector, seleccionar el elemento mínimo del *array* sobre el que multiplicará por -1.

[1] El algoritmo debe devolver el vector óptimo, **no** una aproximación (esto es el vector con suma **máxima** tras *K* transformaciones). Un algoritmo que no encuentre siempre el vector óptimo conlleva una puntuación de 0 en el problema.

[2] El algoritmo debe tener complejidad en tiempo O(*N*·*K*) con respecto el tamaño del vector. Una complejidad mayor implica una puntuación de 0 en el problema.

b) Implementar el algoritmo en Java

```java
int[] vectorTransformado(int[] vector, int K) {
    int[] resultado = new int [vector.length];
    for (int i = 0; i < vector.length; i++)
        resultado[i] = vector[i];
    for (int i = 0; i < K; i++) {
        int posMin = 0;
        for (int j = 0; j < resultado.length; j++) {
            if (resultado[j] < resultado[posMin]) posMin = j;
        }
        resultado[posMin] = -resultado[posMin];
    }
    return resultado;
}
```