Examen de Algorítmica y Complejidad (Plan 2014) **11 de enero de 2023**

Nº matrícula:_____________**Nombre:** ______________________________________

Apellidos: ______________________

Problema 1 (5 puntos). Dado un array de números enteros se quiere reordenar para que todos los números pares queden a la izquierda. Ejemplo: dado el siguiente array:

0	1	2	3	4
-12	**5**	**8**	**9**	**0**

Obtendríamos:

0	1	2	3	4
-12	**8**	**0**	**9**	**5**

a) (4 puntos) Implementar un algoritmo en Java, basado en el ***esquema*** de ***Divide y Vencerás*** con complejidad en el peor caso ***O(NlogN)***[1] que ofrezca esta funcionalidad **sin usar estructuras auxiliares** del tamaño de array. El algoritmo tendrá la siguiente cabecera:

```
void pares(int[] array)
```

```java
public void pares(int[] array) {
        paresAux(array, 0, array.length - 1);
}
public void paresAux(int[] array, int i0, int iN) {
        if (i0 == iN) return;
        else {
            int k = (i0 + iN) / 2;
            paresAux(array, i0, k);
            paresAux(array, k + 1, iN);
            merge(array, i0, k, iN);
        }
}

private void merge(int[] array, int i0, int k, int iN) {
    int l=i0;
    int r=iN;
    // posicionamos l en el primer elemento impar de la mitad izquierda
    while ((l<=k) && (array[l]%2==0)) l++;
    // posicionamos r en el último elemento par de la mitad derecha
    while ((r>k) && (array[r]%2!=0)) r--;
    // mientras haya elementos en la mitad izquierda (que nos hemos asegurado de que
       sean impares) y en la mitad derecha (que nos hemos asegurado de que sean
       pares), los vamos intercambiando, incrementamos l y decrementamos r.
    while ((l<=k) && (r>k)){
        int aux= array[l];
        array[l]=array[r];
        array[r]=aux;
        l++; r--;
```

1 Desarrollar un algoritmo que no esté basado en la estrategia de **Divide y Vencerás**, que no cumpla la complejidad en el peor caso **O(NlogN)**, o use **estructuras auxiliares dependientes de N** conllevará una puntuación de 0 en el ejercicio.

```
    }
}
```

a) (1 punto) Justifica que la complejidad del algoritmo desarrollado en el apartado anterior para el caso peor es de ***O(NlogN)***.

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

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

Esta ecuación es del tipo: $T(N) = p\ T(N/q) + f(N)$, 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 2° del Teorema maestro ($a=\log_q(p)$), por lo que la complejidad del algoritmo es: $T(N) \in O(N^{\log q(p)} \log N) = O(N1\log N) = O(N\log N)$