

## Trabajo práctico N° 2 - Recursión y ordenamiento

### Primera parte

#### Ejercicio 1

Implemente un algoritmo recursivo que determine si un arreglo de tamaño N está ordenado y responda:

1. ¿Qué complejidad O tiene? (La complejidad en el peor caso)
2. ¿Trae algún problema hacerlo recursivo? ¿Cuál?
3. ¿Qué cambiaría si la estructura fuera una lista en lugar de un arreglo?

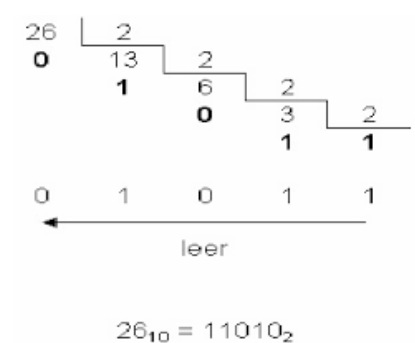
#### Ejercicio 2

Implemente un algoritmo recursivo para buscar un elemento en un arreglo ordenado ascendentemente.

#### Ejercicio 3

Implemente un algoritmo recursivo que convierta un número en notación decimal a su equivalente en notación binaria.

Recordatorio, por ejemplo convertir el 26 a binario:



#### Ejercicio 4

Implemente un algoritmo recursivo que presente los primeros N términos de la secuencia de Fibonacci.

Por ej. los 6 primeros términos son: 0 1 1 2 3 5

### Ejercicio 5

Dado un arreglo ordenado de números distintos  $A$  se desea construir un algoritmo que determine si alguno de los elementos de dicho arreglo contiene un valor igual a la posición en la cuál se encuentra, es decir,  $A[i] = i$ .

1. Construir un algoritmo recursivo que responda a dicha verificación.
2. Mostrar la pila de ejecución para la invocación `algoritmo([-3, -1, 0, 2, 4, 6, 10])`

### Ejercicio 6

Implemente un algoritmo de ordenamiento por selección en un arreglo.

Implemente un algoritmo de ordenamiento por burbujeo en un arreglo.

- ¿Qué complejidad big-O tienen estos algoritmos?

### Ejercicio 7

Implemente un algoritmo de ordenamiento mergesort para un arreglo de tamaño  $N$ .

Implemente un algoritmo de ordenamiento quicksort para un arreglo de tamaño  $N$ .

- ¿Cuál es su complejidad en el peor caso?
- ¿Cuál es su complejidad promedio?

### Ejercicio 8

Investigar cómo está implementado el algoritmo `Collection.Sort` de Java y qué características deben cumplir los elementos almacenados dentro de la colección a ordenar.