	<b>UNIVERSIDAD EAFIT</b> <b>ESCUELA DE INGENIERÍA</b> <b>DEPARTAMENTO DE INFORMÁTICA Y SISTEMAS</b>	<b>Código: ST245</b>
		<b>Estructura de Datos 1</b>

## Laboratorio Nro. 1: Recursion

**Esteban Quintero Marulanda**

Universidad Eafit  
Medellín, Colombia  
equinterom@eafit.edu.co

**Nicolas Roldan Ramirez**

Universidad Eafit  
Medellín, Colombia  
nroldanr@eafit.edu.co

### 2) Ejercicios en línea

#### 2.3

primero verificamos que start sea una posición válida en el arreglo, luego verificamos que en la posición en la que estamos sea divisible por 5 para ver si se incluye en el grupo, y a la vez verificamos que no esté seguido de un 1, posteriormente con los condicionales se busca cuales cumplen para dar el resultado pedido, en este caso llamado target, si es posible retornamos true, de lo contrario, false.

#### 2.4

#### Recursion 1

factorial:

$$T(n) = C + T(n-1)$$

$$T(n) = C' + C*n$$

$$T(n) = O(C' + C*n)$$

$$T(n) = O(n)$$

bunnyEars:

$$T(n) = C + T(n-1)$$

$$T(n) = C' + C*n$$

$$T(n) = O(C' + C*n)$$

$$T(n) = O(n)$$

fibonacci:

**DOCENTE MAURICIO TORO BERMÚDEZ**

Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627

Correo: mtorobe@eafit.edu.co

$$T(n) = C + T(n-1) + T(n-2)$$

$$T(n) = C' + C \cdot 2^n$$

$$T(n) = O(C' + C \cdot 2^n)$$

$$T(n) = O(2^n)$$

bunnyEars2:

$$T(n) = C + T(n-1)$$

$$T(n) = C' + C \cdot n$$

$$T(n) = O(C' + C \cdot n)$$

$$T(n) = O(n)$$

triangle:

$$T(n) = C + T(n-1)$$

$$T(n) = C' + C \cdot n$$

$$T(n) = O(C' + C \cdot n)$$

$$T(n) = O(n)$$

## Recursion 2

groupSum5:

$$T(n) = C + 2T(n-1)$$

$$T(n) = C' \cdot (2^{n-1}) + C(2^n - 1)$$

$$T(n) = O(C' \cdot (2^{n-1}) + C(2^n - 1))$$

$$T(n) = O(2^n)$$

groupSum6:

$$T(n) = C + 2T(n-1)$$

$$T(n) = C' \cdot (2^{n-1}) + C(2^n - 1)$$

$$T(n) = O(C' \cdot (2^{n-1}) + C(2^n - 1))$$

$$T(n) = O(2^n)$$

groupNoAdj:

$$T(n) = C + T(n-1) + T(n-2)$$

$$T(n) = C' + C \cdot 2^n$$

$$T(n) = O(C' + C \cdot 2^n)$$

$$T(n) = O(2^n)$$

groupSumClump:

DOCENTE MAURICIO TORO BERMÚDEZ

Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627

Correo: mtorobe@eafit.edu.co

$$\begin{aligned}T(n) &= C + 2T(n-1) \\T(n) &= C' * (2^{n-1}) + C(2^n - 1) \\T(n) &= O(C' * (2^{n-1}) + C(2^n - 1)) \\T(n) &= O(2^n)\end{aligned}$$

splitArray:


$$\begin{aligned}T(n) &= C + 2T(n-1) \\T(n) &= C' * (2^{n-1}) + C(2^n - 1) \\T(n) &= O(C' * (2^{n-1}) + C(2^n - 1)) \\T(n) &= O(2^n)\end{aligned}$$

## 2.5

La  $n$  y la  $m$  o demás variables asignadas para la misma connotación significan una variable de entrada de dato, siendo  $n$  1 sola,  $n$  y  $m$  cuando son 2, y así respectivamente. Las variables anteriores representan la longitud del problema que se le va a ingresar a cada método, afectando de manera directa la complejidad del mismo.

### 3) Simulacro de preguntas de sustentación de Proyectos

1. El Stack Overflow es un error que ocurre cuando se llena la memoria destinada a almacenar datos, esto quiere decir que se produce una cantidad tan grande de datos almacenados en variables en la memoria, que se llena el cupo que está destinado para esto.
2. 70, además de que da el error Stack Overflow, causa una ineficiencia porque al hacerlo con este tamaño se demoró una cantidad de tiempo muy grande, por lo que era poco efectivo intentarlo con arreglos de mayor tamaño, puesto que sería imposible graficarlo. Por este motivo y por el StackOverflow es que no se puede ejecutar Fibonacci con 1 millón.
3. Una solución muy favorable para calcular Fibonacci con valores más grandes es hacer un proceso de memorización en la memoria para evitar el error de Stack Overflow, el cual consiste en guardar los valores de Fibonacci requeridos para la realización de otros, y de esta manera, no tener que hacer el cálculo completo con otros valores que ya se tendrían, sino que se podrían recuperar directamente desde la memoria.

	UNIVERSIDAD EAFIT ESCUELA DE INGENIERÍA DEPARTAMENTO DE INFORMÁTICA Y SISTEMAS	Código: ST245
		Estructura de Datos 1

4. Los problemas presentados en CodingBat tiene una complejidad de bajo nivel, lo cual los hace muy eficientes, en este caso los de recursión 1, en cambio los de recursión 2 son problemas de más alta complejidad lo cual los hace más difíciles de comprender, son más lentos y requieren de más habilidad para su realización.

#### 4) Simulacro de Parcial

1. Start+1, nums, target
2. b
3. 1. (n-a, b, c, a)  
2. solucionar(n-b, c, a, b), solucionar(n-c, a, b, c)  
3. solucionar(n-a, b, c, a), solucionar(n, a, b, c)
4. e
5. 5.1: return n, n-1, n-2 5.2: b
6. SumaAux(n,i+2), SumaAux(n,i+1)
7. (S,i+1,t-s[i]), (S,i+1,t)