

## Taller en Sala 4

### Complejidad de Algoritmos Recursivos



**Objetivo:** 1. Realizar estudios empíricos para validar una hipótesis sobre el comportamiento en tiempo de ejecución de un algoritmo con varios tamaños de problema. 2. Usar ecuaciones de recurrencia para determinar la complejidad en tiempo de algoritmos definidos de forma recursiva. 3. Usar la notación  $O$  para encontrar formalmente la complejidad asintótica en tiempo de algoritmos



**Consideraciones:** Lean y verifiquen las consideraciones de entrega,



Trabajo en Parejas



Mañana, plazo de entrega



Docente entrega plantilla de código en GitHub



Sí .cpp, .py o .java



No .zip, .txt, html o .doc



Alumnos entregan código sin comprimir GitHub



En la carpeta Github del curso, hay un código iniciado y un código de pruebas (tests) que pueden explorar para solucionar los ejercicios



**Estructura del documento:** a) Datos de *vida real*, b) *Introducción* a un problema, c) Problema a resolver, d) Ayudas. Identifiquen esos elementos así:

a)



b)



c)



d)



**PhD. Mauricio Toro Bermúdez**

Docente | Escuela de Ingeniería | Informática y Sistemas  
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627  
Tel: (+57) (4) 261 95 00 Ext. 9473


**UNIVERSIDAD  
EAFIT**

**Acreditación  
Institucional**  
Renovación  
2018 - 2026  
Resolución MEN 2158 de 2018

## Ejercicios a resolver

- 1** Las iniciativas de la informática ecológica (*green computing*) tienen como uno de sus objetos reducir los efectos del calentamiento de la Tierra mediante la maximización de la eficiencia energética de un software. Un software está compuesto por algoritmos. Para medir la eficiencia de un algoritmo, se analiza el número de recursos (como el tiempo y el almacenamiento) necesarios para ejecutarlos. La eficiencia de un algoritmo se establece como una función que relaciona la longitud de la entrada con el número de pasos (complejidad temporal) que ejecuta el algoritmo. A medida que la eficiencia de un algoritmo aumenta, un menor número de pasos involucrados en el cómputo resultará en un ahorro de energía eléctrica y, por lo tanto, contribuirá a la informática ecológica.



- **Implementen un algoritmo que calcule el máximo de los elementos de un arreglo de forma recursiva.**
- **Calculen la complejidad asintótica para el peor de los casos, tome tiempos para 20 tamaños del problema diferentes, generen una gráfica y analicen los resultados.** ¿La teoría (notación asintótica) corresponde a lo encontrado de forma experimental al tomar los tiempos de la implementación?
-  **Utiliza los conjuntos de datos que se encuentran en la carpeta *datasets*, en Github, para probar tu algoritmo.**

**PhD. Mauricio Toro Bermúdez**

Docente | Escuela de Ingeniería | Informática y Sistemas  
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627  
Tel: (+57) (4) 261 95 00 Ext. 9473

**UNIVERSIDAD  
EAFIT®**

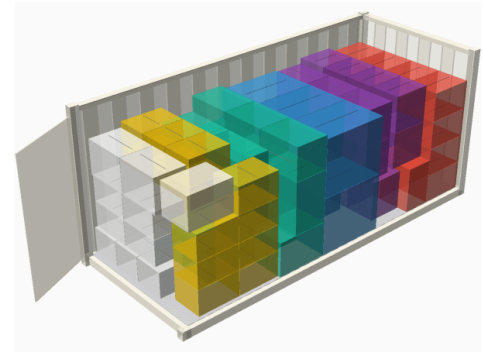
**Acreditación  
Institucional**  
Renovación  
2018 - 2026  
Resolución MEN 2158 de 2018

## ESTRUCTURA DE DATOS 1

### Código ST0245

**2** Puerto Antioquia en Urabá es un proyecto en curso que traerá mucho desarrollo a la región. Su construcción está prevista para terminar a finales de 2020.

Por sus características y ubicación geográfica, Puerto Antioquia podría convertirse en el centro logístico de mayor importancia del país. Un problema que tendrán, una vez inicie la operación, es encontrar la forma óptima de empacar la mercancía en los contenedores.



Una versión simplificada de este problema es, dados los volúmenes de todos los objetos, encontrar un subgrupo de esos volúmenes cuya suma sea igual al volumen máximo del contenedor.

► **Implementen un algoritmo para encontrar si existe un subgrupo de volúmenes cuya suma sea igual a un volumen máximo constante.**

► **Calculen su complejidad asintótica para el peor de los casos, tome tiempos para 20 tamaños del problema diferentes, genere una gráfica y analice los resultados.** ¿La teoría (notación asintótica) corresponde a lo encontrado de forma experimental al tomar los tiempos de la implementación?



En la vida real, la serie de Fibonacci se utiliza para diseñar la distribución de objetos en los videojuegos de tal forma que las escenas cumplan con una cierta estética determinada por la proporción aurea de Fibonacci. Así: <http://bit.ly/2hIQqe1>

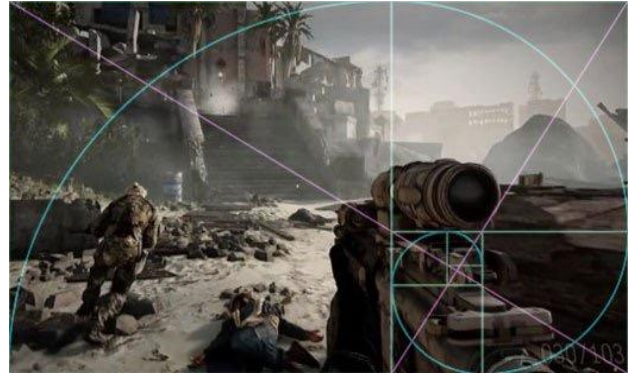
**PhD. Mauricio Toro Bermúdez**

Docente | Escuela de Ingeniería | Informática y Sistemas  
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627  
Tel: (+57) (4) 261 95 00 Ext. 9473

**UNIVERSIDAD**  
**EAFIT**

**Acreditación**  
**Institucional**  
**Renovación**  
**2018 - 2026**  
Resolución MEN 2158 de 2018

**3** [Ejercicio Opcional] En muchos videojuegos, la serie de Fibonacci se utiliza para lograr una estética agradable en el diseño de una escena.



► Implementen un algoritmo que calcule el valor  $n$ -ésimo de la serie de Fibonacci recursivamente.

► Calculen la complejidad asintótica para el peor de los casos, tome tiempos para 20 tamaños del problema diferentes, genere una gráfica y analice los resultados. ¿La teoría (notación asintótica) corresponde a lo encontrado de forma experimental al tomar los tiempos de la implementación?

**PhD. Mauricio Toro Bermúdez**

Docente | Escuela de Ingeniería | Informática y Sistemas  
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627  
Tel: (+57) (4) 261 95 00 Ext. 9473

**UNIVERSIDAD  
EAFIT®**

**Acreditación  
Institucional**  
Renovación  
2018 - 2026  
Resolución MEN 2158 de 2018

# Ayudas para resolver los Ejercicios

Para todos los ejercicios.....	<u>Pág. 5</u>
Ejercicio 1.....	<u>Pág. 7</u>
Ejercicio 2.....	<u>Pág. 7</u>
Ejercicio 3.....	<u>Pág. 8</u>



## Para todos los ejercicios



**Pista 0:** Cálculo de complejidad de los ejercicios del taller

### 1. Suma de los elementos de un arreglo

#### 1.1 Copiar el código en Word

```
private static int suma(int[] a, int i){
    if (i == a.length)
        return 0;
    else
        return a[i] + suma(a,i+1);
}
```

#### 1.2 Identificar quién es el tamaño del problema (llamado también “n”)

El tamaño del problema son los elementos que me falta por sumar en el arreglo.

#### 1.3 Etiquetar cuántas operaciones ejecuta cada línea

```
private static int suma(int[] a, int i){
    if (i == a.length) // constante
        return 0; // constante
    else
        return a[i] + suma(a,i+1); //constante + T(n-1)
}
```

#### 1.4 Escribir la ecuación de recurrencia

$$T(n) = \begin{cases} c_1 & \text{if } n = 1 \\ c_2 + T(n - 1) & \text{if } n > 1 \end{cases}$$

#### 1.5 Resolver la ecuación con Wolfram Alpha

$$T(n) = c_2 + T(n-1)$$

$$T(n) = c_2 * n + c_1$$

**PhD. Mauricio Toro Bermúdez**

Docente | Escuela de Ingeniería | Informática y Sistemas  
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627  
Tel: (+57) (4) 261 95 00 Ext. 9473



## 1.6 Aplicar la notación O a la solución de la ecuación

$T(n)$  es  $O(c_2 \cdot n + c_1)$ , por definición de O

$T(n)$  es  $O(c_2 \cdot n)$ , por Regla de la Suma

$T(n)$  es  $O(n)$ , por Regla del Producto

## 1.7 Explicar en palabras

La complejidad asintótica (es decir, para valores grandes de  $n$ ) para el peor de los casos (es decir, en el que el algoritmo hace más operaciones) para el algoritmo de sumar los elementos de un arreglo recursivamente es  $O(n)$ .

## 2. Suma para ver si es posible alcanzar un valor objetivo

### 2.1 Copiar el código en Word

```
public static boolean SumaGrupo(int start, int[] nums, int target){
    if(start >= nums.length) return target == 0;
    return SumaGrupo(start+1, nums, target-nums[start]) ||
        SumaGrupo(start+1, nums, target);
}
```

### 2.2 Identificar quién es el tamaño del problema (llamado también “n”)

El tamaño del problema es el número de elementos del arreglo.

### 2.3 Etiquetar cuántas operaciones ejecuta cada línea

```
public static boolean SumaGrupo(int start, int[] nums, int target){
    if(start >= nums.length) return target == 0; //constante
    return SumaGrupo(start+1, nums, target-nums[start]) || //constante + T(n-1)
        SumaGrupo(start+1, nums, target); //constante + T(n-1)
}
```

### 2.4 Escribir la ecuación de recurrencia

$$T(n) = \begin{cases} c_1 & \text{if } n = 1 \\ T(n-1) + T(n-1) & \text{if } n > 1 \end{cases}$$

### 2.5 Resolver la ecuación con Wolfram Alpha

**PhD. Mauricio Toro Bermúdez**

Docente | Escuela de Ingeniería | Informática y Sistemas  
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627  
Tel: (+57) (4) 261 95 00 Ext. 9473



## ESTRUCTURA DE DATOS 1

### Código ST0245

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

$$T(n) = c1 \cdot 2^{(n-1)}$$

### 2.6 Aplicar la notación O a la solución de la ecuación

$T(n)$  es  $O(c \cdot 2^{(n-1)})$ , por definición de O  
 $T(n)$  es  $O(2^{(n-1)})$ , por Regla del producto  
 $T(n)$  es  $O(2^n)$ , por Regla de la suma

### 2.7 Explicar en palabras

La complejidad asintótica (es decir, para valores grandes de  $n$ ) para el peor de los casos (es decir, en el que el algoritmo hace más operaciones) para el algoritmo de ver si es posible obtener un valor objetivo entre los elementos, al sumarlos, de un arreglo, recursivamente, es  $O(2^n)$ .



#### Pista 1: Procedimiento sugerido

1. Implementen el algoritmo
2. Identifiquen qué representa el tamaño del problema para cada algoritmo
3. Identifiquen valores apropiados para tamaños del problema
4. Tomen los tiempos para los anteriores algoritmos con 20 tamaños del problema diferentes.
5. Hagan una gráfica en Excel para cada algoritmo y pasarla a Word luego
6. Entreguen el archivo de Word pasado a PDF. Envíen el código y el PDF.



**Pista 2:** Vean la sección 4.2 de la *Guía de Laboratorios* muestra cómo generar arreglos con números aleatorios



**Pista 3:** Vean la sección 4.3 de la *Guía de Laboratorios* muestra cómo aumentar el *heap* en Java y la sección 4.4 muestra cómo aumentar el tamaño de la pila.

**PhD. Mauricio Toro Bermúdez**

Docente | Escuela de Ingeniería | Informática y Sistemas  
 Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627  
 Tel: (+57) (4) 261 95 00 Ext. 9473





## ESTRUCTURA DE DATOS 1

### Código ST0245



**Pista 4:** Vean la sección 4.7 de la *Guía de Laboratorios* muestra cómo tomar el tiempo en Java



**Pista 5:** Vean la sección 4.6 de la *Guía de Laboratorios* muestra cómo cambiar la escala de una gráfica en Excel a escala logarítmica.



### Ejercicio 1



**Pista 1:** *ArrayMax*, calcula el máximo valor de un arreglo de enteros. *A* es el arreglo y *n* es la última posición del arreglo en el primer llamado. *ArrayMax* se llama recursivamente hasta que *n* sea igual a 0.



**Pista 2:** Como un ejemplo *ArrayMax*([1,2,3,8,10,4], 5) retorna 10.



**Pista 3:** El tamaño del problema aquí es el número de elementos del arreglo

```
SubProceso ArrayMax( A, n )
    Definir i, max, temp Como Entero;
    max <- A[n]; // Si n = 0, max <- A[0]
    Si n != 0 Entonces
        temp <- ArrayMax(A, n-1);
        Si temp > max Entonces
            max <- temp;
    Retornar max;
```



### Ejercicio 2

**PhD. Mauricio Toro Bermúdez**

Docente | Escuela de Ingeniería | Informática y Sistemas  
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627  
Tel: (+57) (4) 261 95 00 Ext. 9473



## ESTRUCTURA DE DATOS 1

### Código ST0245



**Pista 1:** El algoritmo *groupSum*, dado un arreglo de enteros, decide si es posible escoger un subconjunto de esos enteros, de tal forma que la suma de los elementos de ese subconjunto sea igual al parámetro *target*. El parámetro *start* funciona como un contador y representa un índice en el arreglo de números *nums*.



**Pista 2:** El tamaño del problema aquí es el número de elementos del arreglo

```
public boolean groupSum(int start, int[] nums, int target) {
    if (start >= nums.length) return target == 0;
    return groupSum(start + 1, nums, target - nums[start])
        || groupSum(start + 1, nums, target);
}
```



### Ejercicio 3



**Pista 1:** El tamaño del problema es el término enésimo (int n).

```
public static long fibonacci(int n) {
    if (n <= 1) return n;
    else return fibonacci(n-1) + fibonacci(n-2);
}
```

**PhD. Mauricio Toro Bermúdez**

Docente | Escuela de Ingeniería | Informática y Sistemas  
 Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627  
 Tel: (+57) (4) 261 95 00 Ext. 9473



# ¿Alguna inquietud?

## CONTACTO

Docente Mauricio Toro Bermúdez

Teléfono: (+57) (4) 261 95 00 Ext. 9473

Correo: mtorobe@eafit.edu.co

Oficina: 19- 627

Agenden una cita dando clic en la pestaña

-*Semana*- de <http://bit.ly/2gzVg10>