

## Laboratorio Nro. 2: Notación O grande

**Jose Andres Carvajal  
Bautista**

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

**Brian Fernando Morales  
Arredondo**

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

**Alejandra Ossa  
Yepes**

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

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

1. En la vida real, para analizar la eficiencia de los algoritmos de forma experimental, debemos probar con problemas de diferente tamaño. El tamaño del problema lo denominamos  $N$ . Como un ejemplo, para un algoritmo de ordenamiento, el tamaño del problema es el tamaño del arreglo. Como otro ejemplo, para calcular la serie de Fibonacci,  $N$  es el número de términos a calcular.

Teniendo en cuenta lo anterior, completen la siguiente tabla con tiempos en milisegundos.

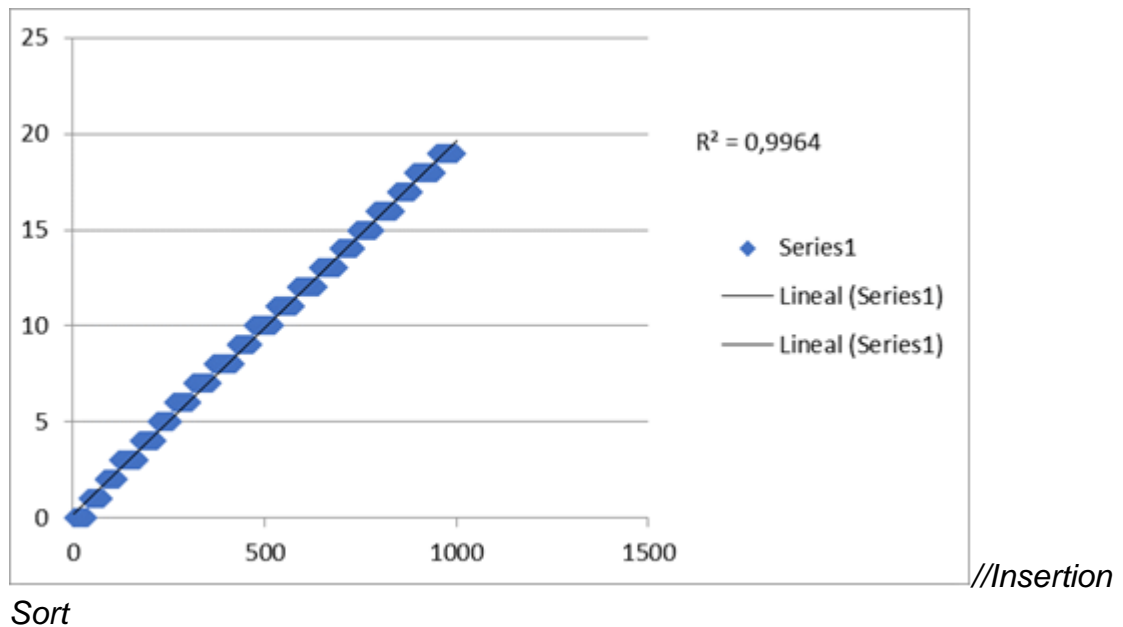
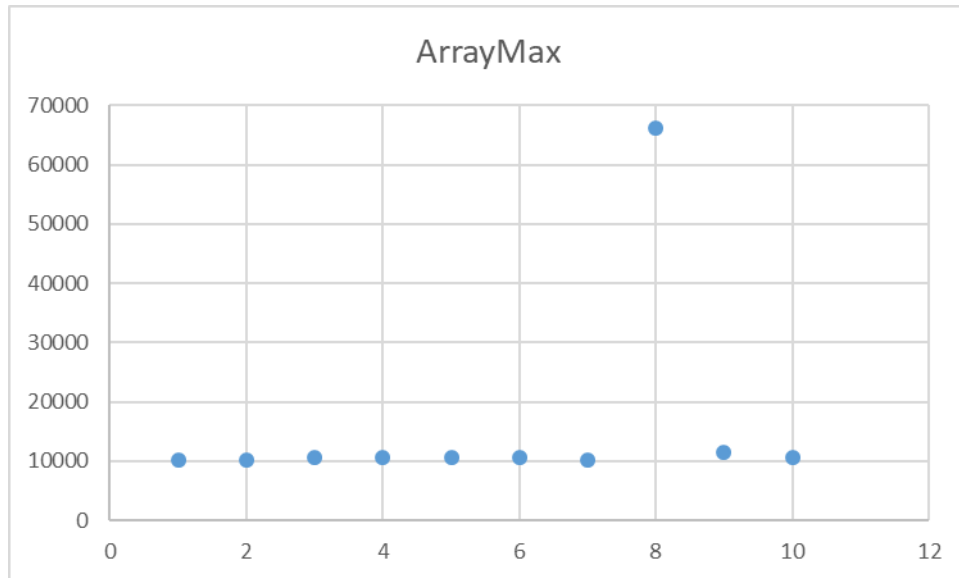
	$N = 100.000$	$N = 1'000.000$	$N = 10'000.000$	$N = 100'000.000$
Array Maximum	0.012656	0.0965	0.1931	1.0012
Insertion Sort	210364572506	21220726128692	Mas de 1 min	Mas de 1 min
Merge Sort	3,19564E+13	3,23996E+15	3,24E+18	

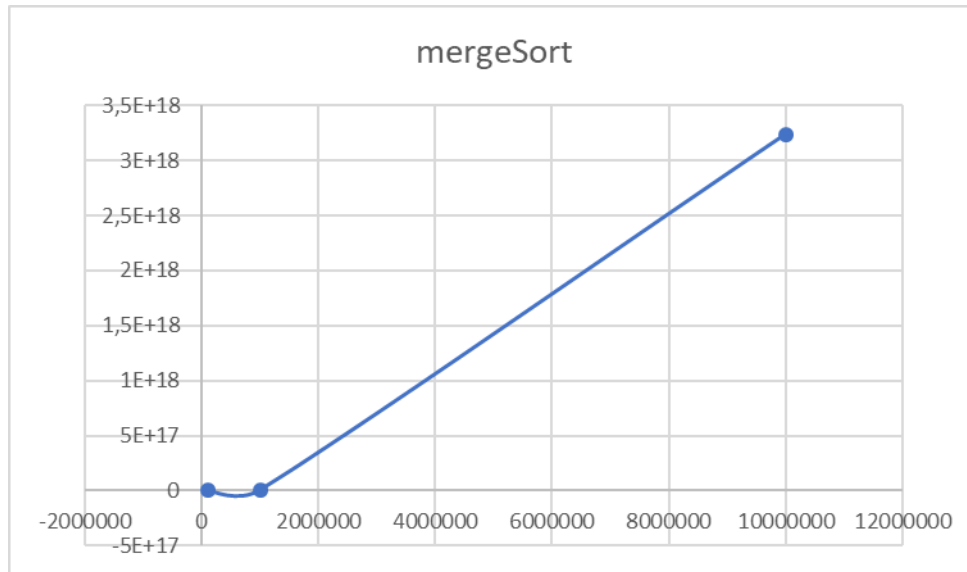
2. Grafiquen los tiempos que tomó en ejecutarse array sum, array maximum, insertion sort y merge sort, para entradas de diferentes tamaños. Si se demora más de un minuto la ejecución, cancela la ejecución y escriba en la tabla "más de 5 minutos". Grafiquen Tamaño de la Entrada ( $N$ ) Vs. Tiempo de Ejecución. Utilicen una escala logarítmica para poder graficar correctamente.

**DOCENTE MAURICIO TORO BERMÚDEZ**

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

Correo: [mtorobe@eafit.edu.co](mailto:mtorobe@eafit.edu.co)





3. ¿Qué concluyen con respecto a los tiempos obtenidos en el laboratorio y los resultados teóricos obtenidos con la notación O?

R/:

Cuando se están midiendo los tiempos dentro del programa al ser los arreglos aleatorios, puede ser posible que alguna vez salgan valores que no sean tan complejos de ordenar, los cuales generaría que el peor caso no suceda, esto puede ocurrir en alguna ejecución del programa, el cual su complejidad se vería afectada a la hora de graficarse y tal vez no saldría el resultado deseado.

4. Teniendo en cuenta lo anterior, ¿Qué sucede con Insertion Sort para valores grandes de N?

R/:

El comportamiento de insertion sort está bien determinado bajo valores pequeños. El cual su comportamiento es muy estándar y sus tiempos son muy manejables, pero cuando el programa se ejecuta en valores más grandes el programa es inconsistente y poco predecible, debido a que su complejidad esta dada en  $O(n^2)$  entre más grande sea el número más difícil va a ser su ejecución.

5. Teniendo en cuenta lo anterior, ¿Qué sucede con ArraySum para valores grandes de N? ¿Por qué los tiempos no crecen tan rápido como Insertion Sort?

R/:

**DOCENTE MAURICIO TORO BERMÚDEZ**

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

**Correo: [mtorobe@eafit.edu.co](mailto:mtorobe@eafit.edu.co)**

En arrays `sum` suma todos los elementos de un arreglo y los elementos no crecen tan rápido como en `insertion sort`, debido a que solo debe coger cada elemento del arreglo, del cual ya sabemos su tamaño, en vez el `insertsort` trabaja recorriendo por partes del arreglo y comparando, el cual hace que sus tiempos crezcan más rápido.

6. *Teniendo en cuenta lo anterior, ¿Qué tan eficiente es Merge sort con respecto a Insertion sort para arreglos grandes? ¿Qué tan eficiente es Merge sort con respecto a Insertion sort para arreglos pequeños?*

R/:

Insertion Sort para valores grandes tiene un comportamiento muy estándar, a diferencia de Merge Sort, cuyos valores tienden a ser variados. En cuanto a valores más grandes, Merge Sort nos da valores más predecibles y estándar, a diferencia de Insertion Sort, cuyos valores son completamente variados y crecientes.

7. *Expliquen con sus propias palabras cómo funciona el ejercicio `maxSpan` y ¿por qué?*

R/:

`maxSpan` es un método que recibe un arreglo. Luego analiza si su longitud es mayor que cero. De ser así, crea un ciclo, con un ciclo anidado, esto con el propósito de poder evaluar cuales elementos son iguales en el arreglo. Cuando dos elementos son iguales, ve la diferencia entre estos dos elementos y le suma uno. Si esta diferencia sumada uno es mayor que uno, para el ciclo y retorna uno. De ser todo lo contrario falso, retorna cero.

8. *Calculen la complejidad de los ejercicios en línea, numerales 2.1 y 2.2, y agréguenla al informe PDF*

a) 2.1

1. `has22` →  $O(n)$
2. `countEvans` →  $O(n)$
3. `sum67` →  $O(n^2)$
4. `lucky13` →  $O(n)$
5. `sum13` →  $O(n)$

b) 2.2

1. `maxSpan` →  $O(n^2)$
2. `fix34` →  $O(n^2)$
3. `fix45` →  $O(n^2)$
4. `canBalance` →  $O(n^2)$
5. `linearIn` →  $O(n + m)$

9. Expliquen con sus palabras las variables (qué es 'n', qué es 'm', etc.) del cálculo de complejidad del numeral anterior.

R/:

$n$  es una variable que nos indica cual es la longitud del arreglo insertado en el arreglo. En el caso de que se presente también la variable  $m$ , es debido a que ya no solo la complejidad del tamaño de un arreglo insertado, si no de dos arreglos.

#### 4) Simulacro de Parcial

1. Supongamos que  $P(n,m)$  es una función cuya complejidad asintótica es  $O(n \times m)$  y  $H(n,m)$  es otra función cuya complejidad asintótica es  $O(m \cdot A(n,m))$ , donde  $A(n,m)$  es otra función. ¿Cuál de las siguientes funciones, definitivamente, NO podría ser la complejidad asintótica de  $A(n,m)$  si tenemos en cuenta que  $P(n,m) > H(n,m)$  para todo  $n$  y para todo  $m$ ?

c)  $O(n+m)$

2. Dayla sabe que la complejidad asintótica de la función  $P(n)$  es  $O(\sqrt{n})$ . Ayúdala a Dayla a sacar la complejidad asintótica para la función  $mystery(n,m)$ .

d)  $O(m \times n)$

3. El siguiente algoritmo imprime todos los valores de una matriz. El tamaño de la matriz está definida por los parámetros largo y ancho. Teniendo en cuenta que el largo puede ser como máximo 30, mientras que el ancho puede tomar cualquier valor, ¿cuál es su complejidad asintótica en el peor de los casos del algoritmo?

b)  $O(\text{ancho})$

4. Sabemos que  $P(n)$  ejecuta  $n^3 + n$  pasos y que  $D(n)$  ejecuta  $n+7$  pasos. ¿Cuál es la complejidad asintótica, en el peor de los casos, de la función  $B(n)$ ?


b)  $O(n^3)$

5. El siguiente algoritmo calcula las tablas de multiplicar del 1 a  $n$ . ¿Cuál es su complejidad asintótica en el peor de los casos?

d)  $O(n^2)$

#### 5. Lectura recomendada (opcional)

- a) Título
- b) Ideas principales
- c) Mapa de Conceptos

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

## 6. Trabajo en Equipo y Progreso Gradual (Opcional)

- a) Actas de reunión
- b) El reporte de cambios en el código