Laboratorio Nro. 2: Notación O grande

Carla Daniela Rendón
Universidad Eafit
Medellín, Colombia
cdrendonb@eafit.edu.co

Sebastián Arboleda
Universidad Eafit
Medellín, Colombia
sarboledab@eafit.edu.co

3) Simulacro de preguntas de sustentación de Proyectos

1. 3.1 De acuerdo a lo realizado en el numeral 1.1, completen la siguiente tabla con tiempos en milisegundos, considerando la versión recursiva de los algoritmos:

   Nota: Debido a que la computadora presenta error al ingresarle arreglos de tamaño tan elevado, usaremos la proporcionalidad para obtener los resultados, ejemplo: Si para resolver un problema con un arreglo de tamaño N=1000, para el caso de ArraySum se tarda un tiempo de: 3185 milisegundos, entonces para un arreglo de tamaño N=100000 se tardará 100 veces el valor anterior.

	N=100.000	N=1.000.000	N=10.000.000	N=100.000.000
ArraySum	**318500**	**3185000**	**31850000**	**318500000**
ArrayMax	**340900**	**3409000**	**34090000**	**340900000**
Fibonacci	**Más de un minuto**	**Más de un minuto**	**Más de un minuto**	**Más de un minuto**

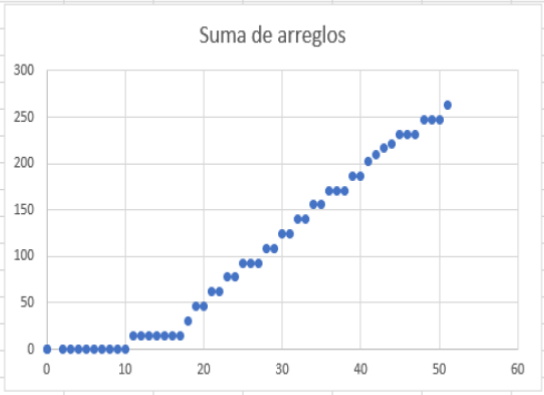
2. Grafiquen los tiempos que tomó en ejecutarse Array Sum, Array Maximum y Fibonacci recursivo, para entradas de diferentes tamaños y para la versión recursiva de los mismos. Si se demora más de un minuto la ejecución, cancela la ejecución y escriba en la tabla "más de 1 minuto".

Fibonacci

valor	tiempo
12	0
14	0
16	1
18	1
20	0
22	1
24	0
26	1
28	3
30	5
32	13
34	37
36	86
38	227
40	625
42	1503
44	4139
46	9955
48	26054
50	66886

Suma de arreglos:

3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0
11	15
12	15
13	15
14	15
15	15
16	15
17	15
18	31
19	46
20	46
21	62



DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co

3. ¿Qué concluyen respecto a los tiempos obtenidos en el numeral 3.1 del presente y los resultados teóricos?

Respuesta: Con respecto a los resultados obtenidos, podemos concluir que el tiempo de ejecución de varía dependiendo de la complejidad del algoritmo. Mientras más grande sea el N, mayor será el tiempo de ejecución.

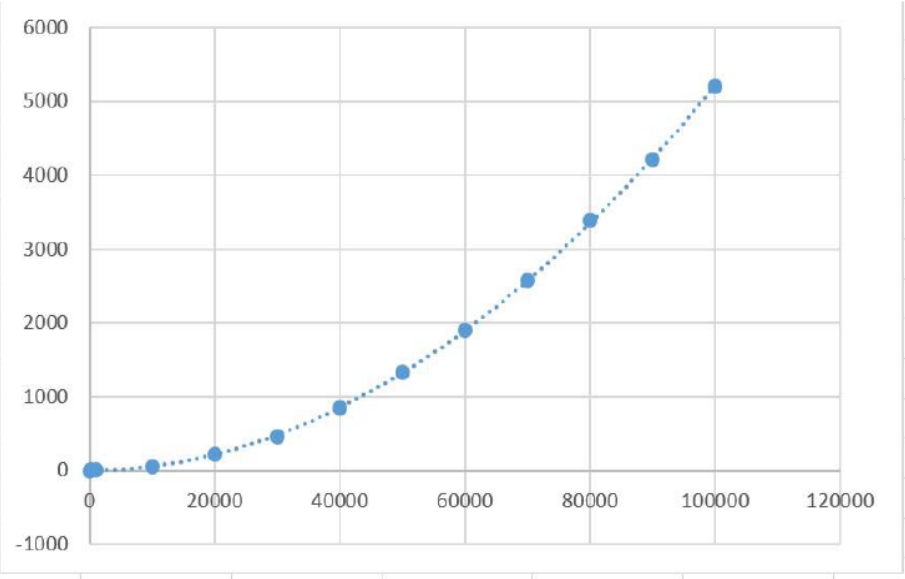
4. Complete la siguiente tabla con tiempos en milisegundos

	N=100000	N=1000000	N=10000000	N=100000000
ArraySum (No recursivo)	265700	2657000	26570000	265700000
ArrayMax (No recursivo)	232200	2322000	23220000	232200000
Insertion Sort	443100	4431000	44310000	443100000
Merge Sort	376700	3767000	37670000	376700000

5. Grafiquen los tiempos que tomó en ejecutarse array sum, array maximum, insertion sort y merge sort.

Ordenamiento por inserción:

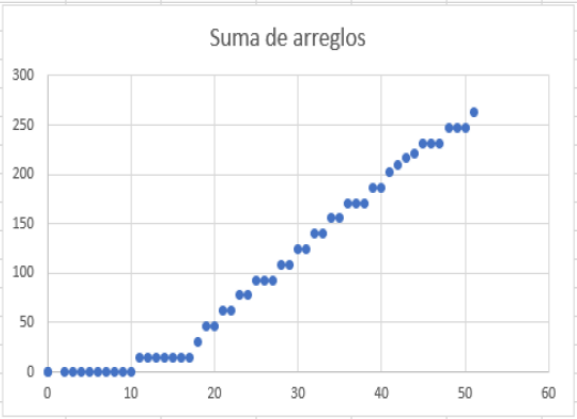
10	0
100	1
1000	2
10000	54
20000	219
30000	462
40000	850
50000	1329
60000	1898
70000	2580
80000	3390
90000	4218
100000	5208



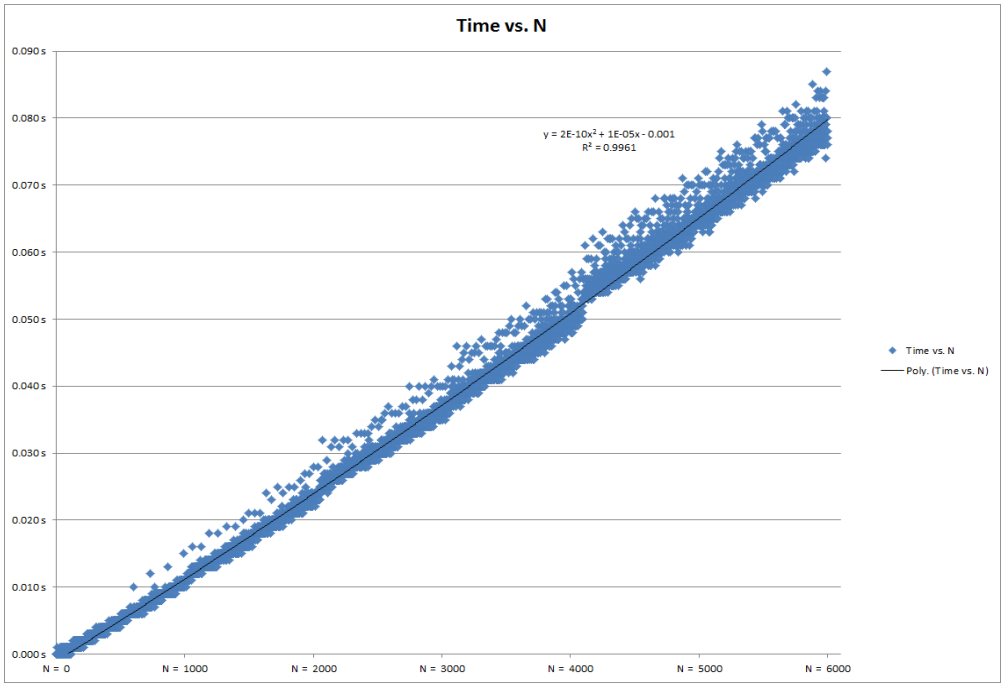
DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co

Suma de arreglos:

3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0
11	15
12	15
13	15
14	15
15	15
16	15
17	15
18	31
19	46
20	46
21	62



MergeSort:

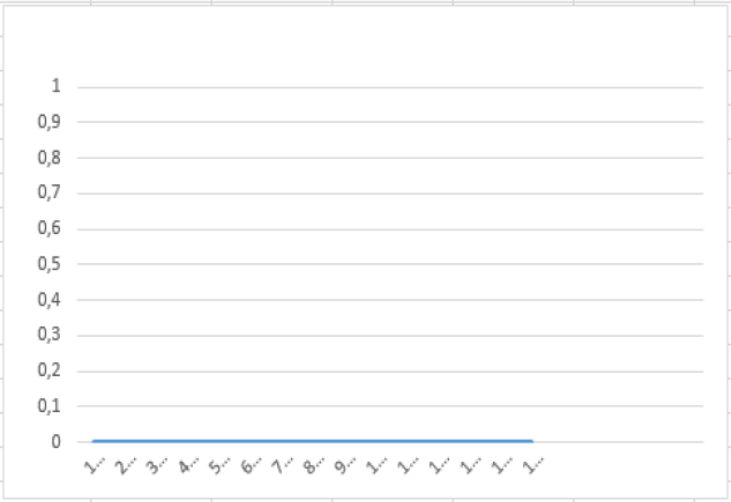


(Gráfica sacada de: https://stackoverflow.com/questions/24728007/time-complexity-of-mergesort-implementation)

DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co

arrayMax

valor	tiempo
10000	0
20000	0
30000	0
40000	0
50000	0
60000	0
70000	0
80000	0
90000	0
100000	0
110000	0
120000	0
130000	0
140000	0
150000	0



6. **¿Qué concluyen con respecto a los tiempos obtenidos en el numeral 3.4 del presente y los resultados teóricos obtenidos?**

Respuesta: Con respecto a los resultados obtenidos, pododemos concluir que el tiempo varía dependiendo del tipo de complejidad del algoritmo, y que cuando se realizan llamados iterativos se tarda menos tiempo en ejecutarse en programa.

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

Insertion Sort es un algoritmo que permite ordenar, este que recurre a una búsqueda binaria en lugar de una búsqueda secuencial para insertar un elemento en la parte de arriba del arreglo, que ya se encuentra ordenado. A medida que los valores de N(cantidad de elementos del arreglo) van aumentando, el tiempo que se demora en ejecutar el Insertion Sort aumenta muy rápidamente.

8. **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?**

Al igual que todos los algoritmos analizados en este laboratorio, ArraySum a medida que aumenta el N, aumenta el tiempo de ejecución. Los tiempos no crecen tan rápido como en insertion Sort debido al tipo de complejidad de este algoritmo.

9. ¿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?

Consideramos que Merge Sort tiende a ser más eficiente que insertion sort para arreglos grandes, ya que le tiempo de ejecución es menor. Para arreglos más pequeños, consideramos más eficiente a Insertion Sort ya que el tiempo de ejecución es menor.

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

Este algoritmo lo que hace es retornar la cantidad de elementos que hay entre el numero más pequeño y más grande de un arreglo.

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

Para los ejercicios del numeral 2.1 tenemos una compledidad de O(n), este es considerado uno de los peores casos. Para los ejercicios del numeral 2.2 tenemos una complejidad de O(n^2) para bubble sort (caso peor o implementación sencilla), Shell sort, quicksort (caso peor).

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

Las variables n y m representan los parámetros de entrada del algoritmo.

4) Simulacro de Parcial

1. *c*
2. d
3. *d*
4. a
5. a
6. a
7.1 T(n)=T(n-1)+c
7.2 O(n)

DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co