ESTRUCTURA DE DATOS 1
Código ST0245

Laboratorio Nro. 2
Complejidad de algoritmos

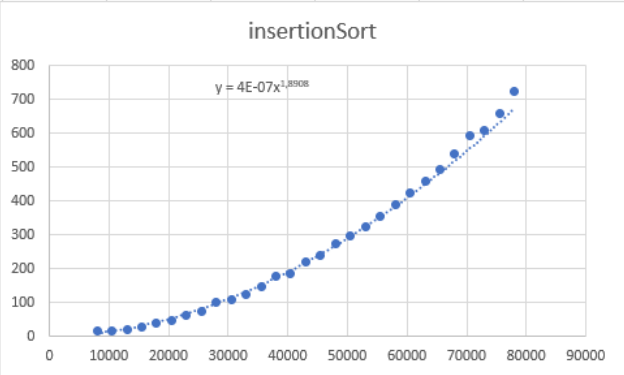
Stiven Yepes
Universidad Eafit
Medellín, Colombia
syepesv@eafit.edu.co

Sara Rodríguez
Universidad Eafit
Medellín, Colombia
srodriguev@eafit.edu.co

1) Simulacro de proyecto, algoritmos de ordenamiento

Insertion Sort

8000	14
10500	15
13000	20
15500	28
18000	39
20500	48
23000	61
25500	74
28000	102
30500	108
33000	124
35500	146
38000	176
40500	185
43000	220
45500	240
48000	273
50500	296
53000	324
55500	355
58000	390
60500	423
63000	459
65500	493
68000	538
70500	592
73000	611
75500	659
78000	723

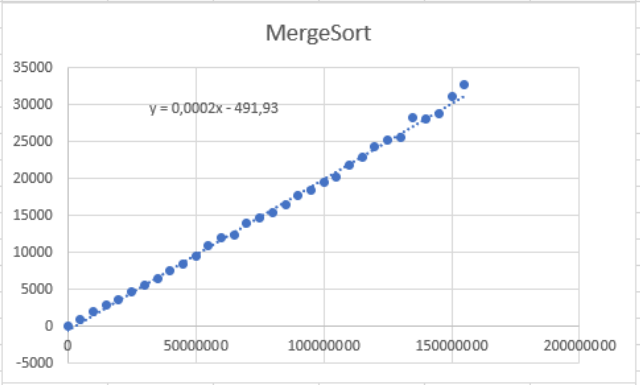


Complejidad = O(n^2)

Merge Sort

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

10000	3
5010000	1016
10010000	1995
15010000	3007
20010000	3721
25010000	4692
30010000	5634
35010000	6551
40010000	7643
45010000	8419
50010000	9458
55010000	10935
60010000	12048
65010000	12456
70010000	13964
75010000	14759
80010000	15495
85010000	16539
90010000	17670
95010000	18473
100010000	19604
105010000	20250
110010000	21797
115010000	22879
120010000	24398
125010000	25225
130010000	25541
135010000	28278
140010000	28058
145010000	28860
150010000	31125
155010000	32810



Complejidad = O(n*log(n))

3) Simulacro de preguntas de sustentación de Proyectos

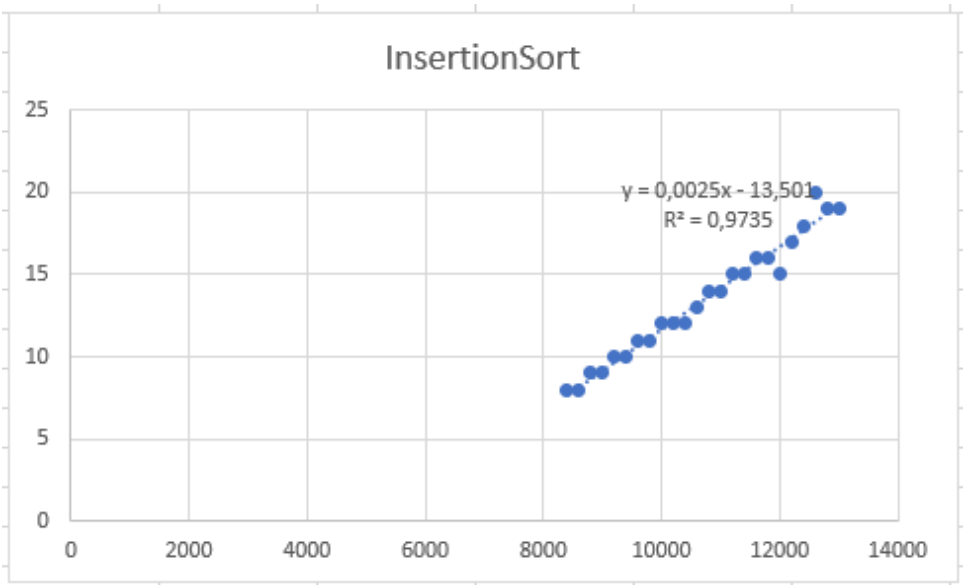
3.1 Tabla anexa en punto 1**.**

3.2 Gráfica para diferentes tamaños del problema

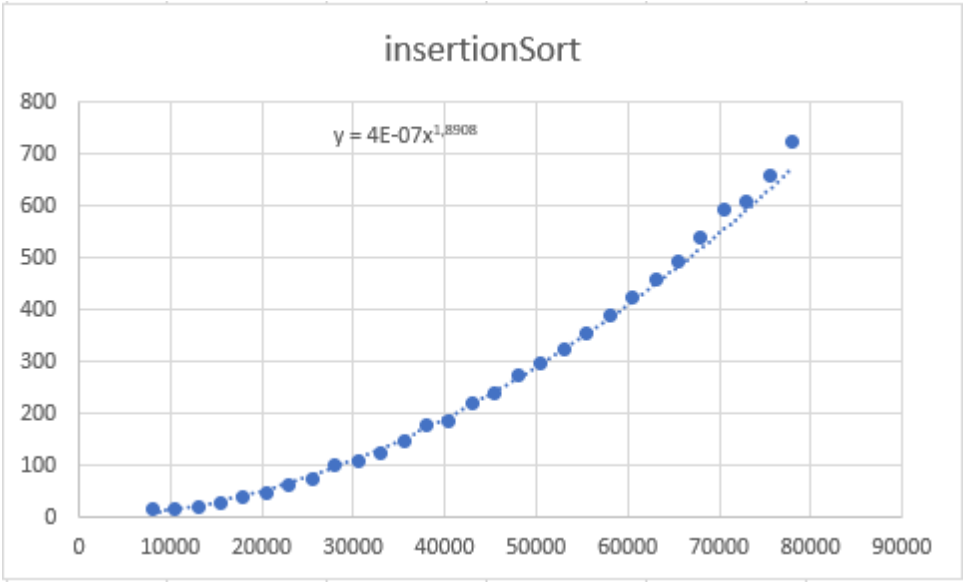
InsertionSort:

Para este algoritmo primero se definieron tamaños de arreglos entre 8000 y 13000 elementos, sin embargo, el resultado que se tenía en la gráfica era algo inconcluso, se veía similar a una recta “pixelada”.

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



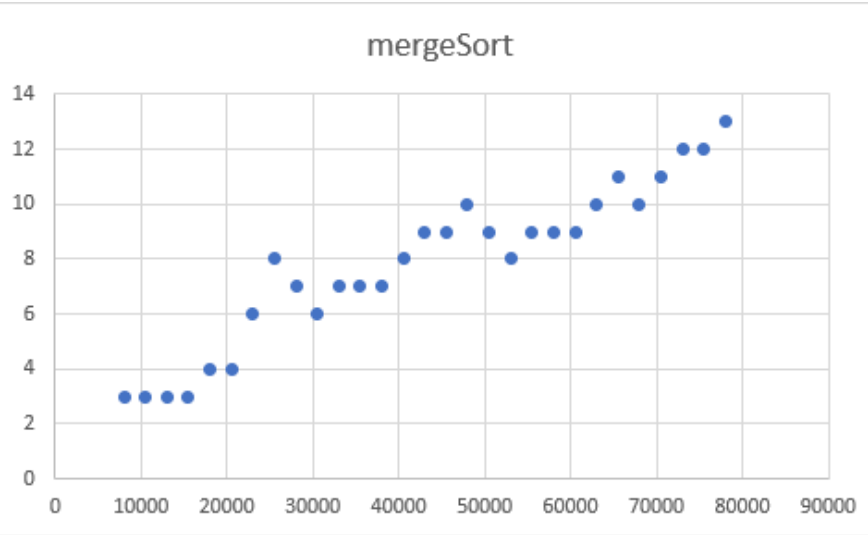
Para tener un resultado más concluyente se amplía el rango de 8000 a 78000 elementos, creando una curva más marcada.



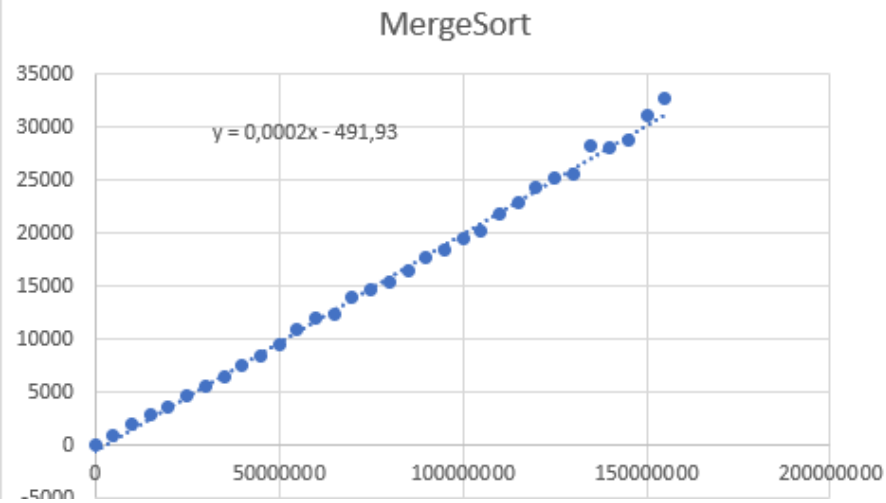
MergeSort:

En la primera medición de mergesort se tomó un rango de 8000 a 78000 elementos en un arreglo, pero aun así el rango de tiempo era algo bajo, entre 3 y 13 ms.

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



De esta manera se aumenta el rango desde 10,000 hasta 155,010,000 elementos a sortear para poder obtener una gráfica más acorde, como la siguiente.



3.3 ¿Qué tan eficiente es merge sort con respecto a insertion sort para arreglos grandes?

Merge-sort para números bastante grandes presenta un mejor desempeño que Insertion-sort, esto se debe a su complejidad, que mezcla una recta (complejidad n) con una función logarítmica (complejidad logn), mucho menores que la presente en insertion sort, que es n^2. Sin embargo si se tuviera O(n) u O(logn) se hablaría de una diferencia en desempeño aún más notable.

La ventaja de Merge-sort se debe gracias a que al “partir” un grupo a la mitad se va avanzando mucho más en el proceso de organización porque se divide el problema en partes

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

más manejables que cuidando y ordenando el lugar de los elementos uno por uno, podría decirse que de manera similar al dicho “divide y vencerás”.

3.4 ¿Es apropiado usar insertion-sort para un videojuego con millones de elementos en una escena y demandas de tiempo real en la renderización?

El render o procesamiento de gráficos en un videojuego es uno de los principales problemas del mercado y uno de los puntos de venta de los PC “gama alta”, ya que se tienen muchas herramientas especializadas como las tarjetas gráficas (GPU) para manejar estas variables, que son bastante complicadas para el CPU y muchas veces requieren una unidad especial que las procese razón que no todos los computadores puedan correr todos los juegos, llamada GPU.

Así que, este tema es uno altamente complejo donde se debe optimizar al máximo. En el mercado actual hay empresas que aparentemente se dedican a desarrollar este tipo de sortings, según lo evidenciado en google (un ejemplo hallado más abajo) por lo cual hay bastante seguridad de que hay muchos algoritmos más eficientes disponibles para esta tarea con complejidades mucho más manejables.

Referencia: Sorting using graphic processors: http://gamma.cs.unc.edu/SORT/
Como se puede ver el sorting es un problema bastante relevante que está en constante búsqueda una mejor solución.

3.5 Para arreglos grandes, ¿en qué caso insertion-sort es más rápido que merge-sort? ¿cómo deben ser los datos para que insertion sort sea más rápido que merge sort?

Los datos deben estar parcialmente ordenados.

Si los datos ya tienen un considerable grado de orden el insertion-sort será más rápido que merge sort porque tendrá que realizar menos operaciones que en el peor escenario posible, mientras que un orden parcial no afecta el funcionamiento del merge al mismo grado, porque igualmente se dividirá en los mismos grupos para hacer la búsqueda y ordenamiento.

3.6

3.7 y 3.8 Calculen la complejidad de los Ejercicios en Línea de los numerales 2.1. Expliquen con sus palabras las variables (qué es ‘n’, qué es ‘m’, etc.) del cálculo de complejidad del ejercicio anterior.

Array2:

- Countevens

Complejidad O(n)
En este problema n representa la necesidad de recorrer un ciclo de n elementos.

- BigDiff

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

Complejidad O(n)
En este problema n representa la necesidad de recorrer un ciclo de n elementos.

- Sum13

Complejidad O(n)
En este problema n representa la necesidad de recorrer un ciclo de n elementos.

- Has22

Complejidad O(n)
En este problema n representa la necesidad de recorrer un ciclo de n elementos.

- Only14

Complejidad O(n)
En este problema n representa la necesidad de recorrer un ciclo de n elementos.

Array3:

- MaxSpan

Complejidad O(n^2)
En este problema el n a la dos representa que se tiene un ciclo for dentro de un ciclo for.

- Fix34

Complejidad O(n^2)
En este problema el n a la dos representa que se tiene un ciclo for dentro de un ciclo for.

- SquareUp

Complejidad O(n^2)
En este problema el n a la dos representa que se tiene un ciclo for dentro de un ciclo for.

- CanBalance

Complejidad O(n) , (O(mn))
En este problema la complejidad exacta son 2n, que se puede abreviar como n, ya que dos es una constante, a veces también se representa como O(mn) donde m es constante, en este caso m sería 2. El 2n representa que se tienen dos ciclos, no anidados, sino una detrás de otro.

- LinearIn

Complejidad O(n)
En este problema n representa recorrer todo un arreglo dentro de un ciclo while

- Fix45

Complejidad O(n) , (O(mn))
En este problema la complejidad exacta son 2n, que se puede abreviar como n, ya que dos es una constante, a veces también se representa como O(mn) donde m es constante, en este

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

caso m sería 2. El 2n representa que se tienen dos ciclos, no anidados, sino una detrás de otro.

4) Simulacro de Parcial

4.1. d) O(1)

4.2 d) O(mxn)

4.3 b) O(ancho)

4.4 b) O(n^3)

4.5 d) T(n)=T(n/10)+c, que es O(logn)
(b) No

4.6 Tardará aproximadamente 10 segundos

4.7 1 y 4 son verdaderas.

4.9 a) O(n^3)

4.10 c) menos de n.log n pasos (genera un error, no hay j)

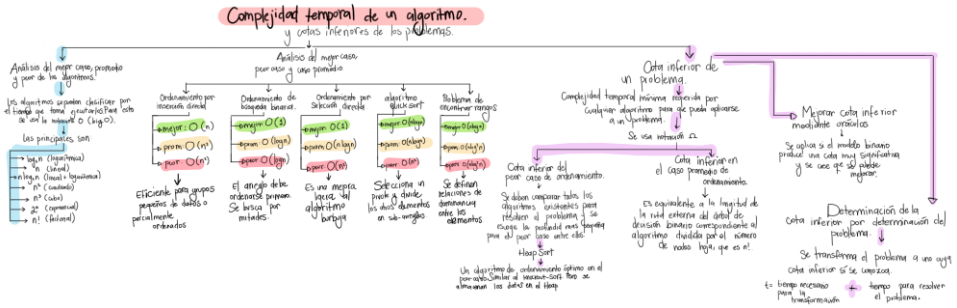
4.11 c) ejecuta T(n)= T(n-1) + T(n-2) + c pasos.

4.12 b) O(m x n x log(n) + n x m^2 + n^2 x log(n) + m^3)

4.13 c) T(n)=2T(n/2) + n

4.14 c) O(m x √(m) + n^3)

5) Lectura recomendada (opcional)



Disponible en tamaño real:

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

https://eafit-my.sharepoint.com/:i:/g/personal/srodriguev_eafit_edu_co/Ebz9CN9KJaBJgC1fFjCdwL0BxMSZgd_LEifeI8o-IixOgg?e=l4wZqq

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