ESTRUCTURA DE DATOS 1
Código ST0245

Laboratorio Nro. 2
Complejidad de algoritmos

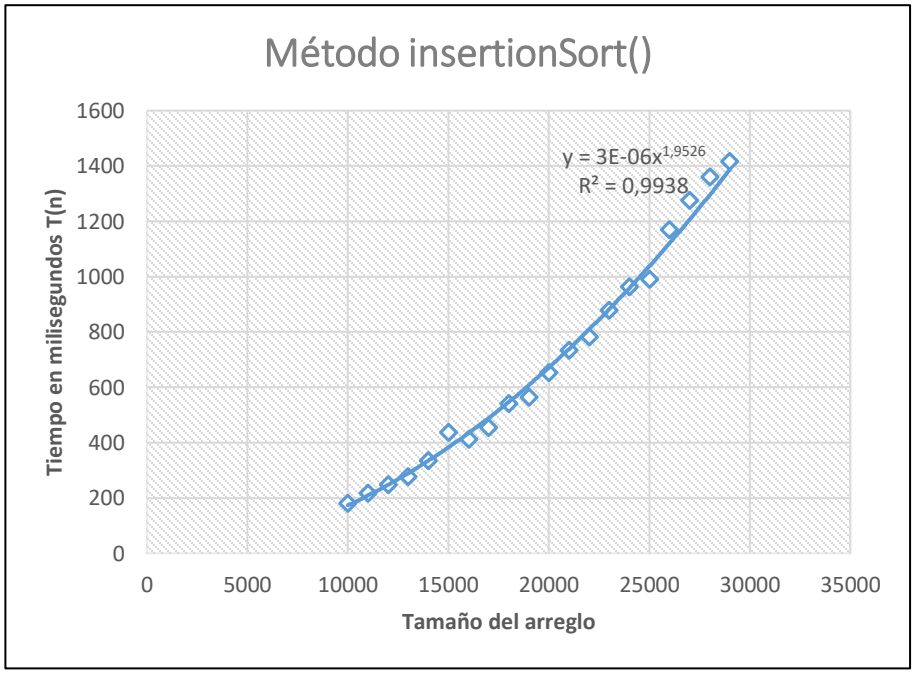
Jacobo Rave Londoño
Universidad Eafit
Medellín, Colombia
jravel@eafit.edu.co

Juan Sebastián Guerra Hernández
Universidad Eafit
Medellín, Colombia
jsguerrah@eafit.edu.co

3) Simulacro de preguntas de sustentación de Proyectos

3.1 OPCIONAL y 3.2 insertionSort()

Tamaño del arreglo n	T(n)
10000	180
11000	217
12000	247
13000	277
14000	335
15000	436
16000	412
17000	453
18000	540
19000	564
20000	652
21000	733
22000	782
23000	878
24000	962
25000	990
26000	1168
27000	1275
28000	1360
29000	1415

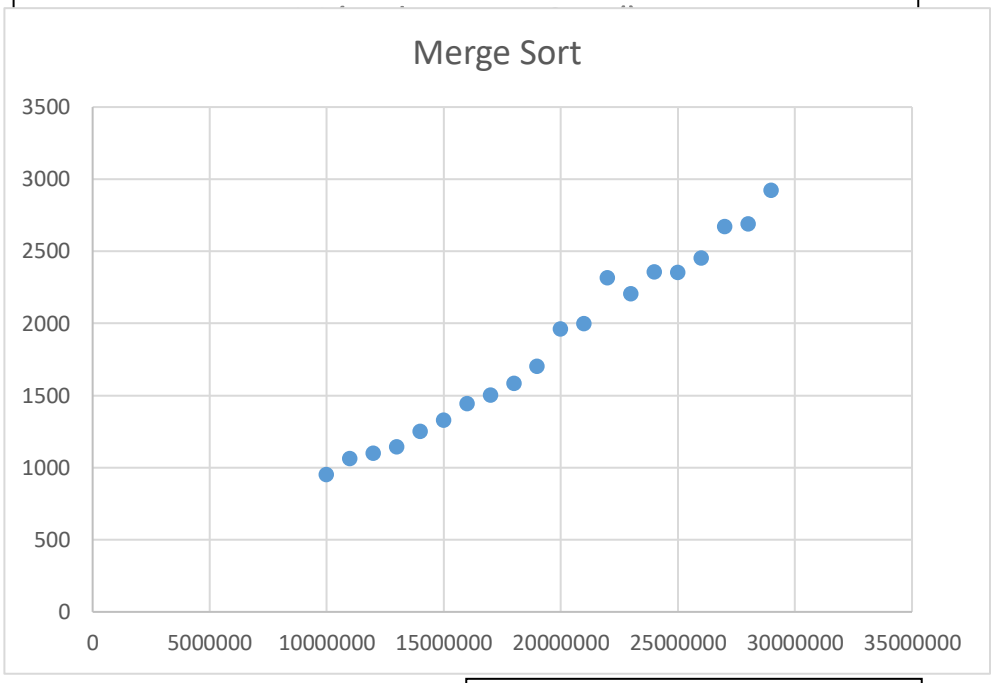


Complejidad asintótica es $O(n^2)$

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

mergeSort()

Tamaño del arreglo n	T(n)
10000000	953
11000000	1061
12000000	1101
13000000	1142
14000000	1251
15000000	1327
16000000	1444
17000000	1501
18000000	1582
19000000	1701
20000000	1961
21000000	1997
22000000	2316
23000000	2204
24000000	2356
25000000	2351
26000000	2453
27000000	2672
28000000	2688
29000000	2923



Complejidad asintótica es O(nLog(n))

3.3 Es recomendable utilizar una complejidad O(n) para el procesamiento de millones de datos a comparación de una O(nLog(n)); el procesamiento de los datos se haría en muy pocos segundos, lo que involucra que pueda ser utilizado en videojuegos. Sin embargo, una complejidad así puede generar delay para la jugabilidad en línea, por lo que si se cuenta con más opciones de ordenamiento con menor complejidad se pueden sobreponer sobre el uso de insertionSort.

3.4 El algoritmo mergeSort tiene dos métodos principales:

- El primero se va a encargar de dividir en 2 el arreglo cuantas veces sea necesario para descomponerlo. De esta forma, se va a repetir Log(n) veces hasta tenerlo descompuesto.

- Nuestro segundo método cumple la función de organizar según sus valores. Este proceso se va a repetir n veces en el peor de los casos.

El segundo método al ser llamado dentro del primero, hace que las complejidades entre ambos se multipliquen, dando con un resultado de O(nLog(n)).

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

3.4 OPCIONAL

El método ***maxSpan*** funciona de la siguiente manera:

1. Se inicializan dos variables internas *span* y *aux* donde iremos guardando nuestro progreso.
2. El primer *for* cumple la función de recorrer el arreglo para la comparación del primer número. Por otro lado, el segundo *for* nos traerá nuestro segundo número a comparar. Basado en cómo están organizados, lo que se va a hacer es que se va a comparar el primer elemento del arreglo con todos los elementos del arreglo, luego se intenta con el segundo y, de esta forma se va a lograr la comparación de cada elemento con los demás del arreglo.
3. Nuestro *if* se va a basar en la veracidad de la condición de igualdad entre nuestro primer y segundo número a comparar:
    a. Si son iguales entonces en *aux* calculamos el valor absoluto de i-j (recordar que tanto i como j son iteradores de posición, por lo que su resta nos dará la distancia entre dichas posiciones). El +1 es importante porque los arreglos utilizan por su definición la posición 0 como inicial.
    b. Luego de esto verificamos si esa diferencia de posiciones es mayor que alguna diferencia de posición anterior. Dado el caso que sea cierto, esta será nuestro nuevo *span*.
    c. Si la igualdad de los dos números a comparar es falsa, se procede a comparar con el siguiente elemento.

Gracias a los ciclos el proceso se va a repetir hasta comparar todos los elementos, encontrando la diferencia de posición más larga entre números iguales del arreglo (*span*).

3.5 y 3.6
Array-2

	Complejidad	***Entradas***
countEvens()	O(n)	n es el tamaño del arreglo
bigDiff()	O(n)	n es el tamaño del arreglo
lucky13()	O(n)	n es el tamaño del arreglo
no14()	O(n)	n es el tamaño del arreglo
only14()	O(n)	n es el tamaño del arreglo

Array-3

	Complejidad	***Entradas***
maxSpan()	$O(n^2)$	n es el tamaño del arreglo
maxMirror()	$O(n^3)$	n es el tamaño del arreglo
countClumps()	O(n)	n es el tamaño del arreglo
fix34()	$O(n^2)$	n es el tamaño del arreglo
fix45()	$O(n^2)$	n es el tamaño del arreglo

4) Simulacro de Parcial

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

4.1 OPCIONAL C
4.2 D
4.3 OPCIONAL B
4.4 OPCIONAL A
4.5.1 D
4.5.2 B
4.6 100s
4.7 1-3-4
4.8 OPCIONAL A
4.9 A
4.10 OPCIONAL C
4.11 OPCIONAL C
4.12 OPCIONAL B
4.13 OPCIONAL C
4.14 A y C

5) Lectura recomendada (opcional)

Mapa conceptual

6) Trabajo en Equipo y Progreso Gradual (Opcional)

6.1 *Actas de reunión*
6.2 *El reporte de cambios en el código*
6.3 *El reporte de cambios del informe de laboratorio*

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