ESTRUCTURA DE DATOS 1
Código ST0245

Laboratorio Nro. 2
Complejidad de algoritmos

Diego Alexander Múnera Tobon
Universidad Eafit
Medellín, Colombia
damunerat@eafit.edu.co

María Antonia Velasquez
Universidad Eafit
Medellín, Colombia
mavelasqur@eafit.edu.co

1.1) Complejidad Insertion Sort:

En vista de que el algoritmo para tamaños del arreglo[n] pequeños toma un tiempo casi despreciable (0.0 segundos), se calculará la complejidad para el peor caso, o sea cuando el array está organizado de menor a mayor.

T(n) = C1 * n
T(n) = (C2 + C3) *(N-1)
T(n) = C4*(N-1)(N)/2
T(n) = (C5 + C6)*((N-1) (N)/2-1)
T(n) = C8*(N-1)

T(n) = C*(n^2)
O(n) = O(n^2)

Complejidad Merge Sort:

Tiempo de ejecución para un array de 9 elementos en desorden = 0.00901055 segundos.

T(n) = C1/2 + C2/2
T(n) = 2T(n/2)
O(n) = nlog(n)

3) Simulacro de preguntas de sustentación de Proyectos

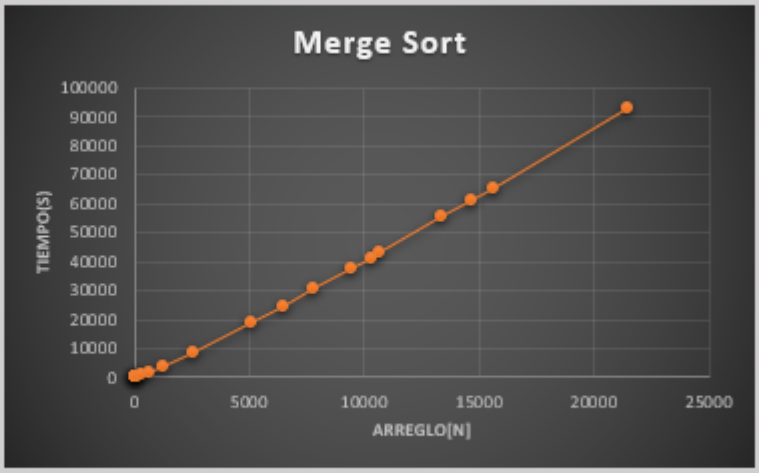
3.1 y 3.2
Insertion Sort:

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

Arreglo[n]	Tiempo(s)
5	25
10	100
20	400
40	1600
80	6400
160	25600
320	102400
640	409600
1280	1638400
2560	6553600
5120	2.6E+07
6480	4.2E+07
7820	6.1E+07
9460	8.9E+07
10320	1.1E+08
10640	1.1E+08
10860	1.2E+08
10980	1.2E+08
11000	1.2E+08
11200	1.3E+08

Merge Sort:

Arreglo[n]	Tiempo(s)
5	3.49485
10	10
20	26.0206
40	64.0824
80	152.247
160	352.659
320	801.648
640	1795.96
1280	3977.23
2560	8725.09
5120	18991.5
6480	24699
7820	30444.9
9460	37611.9
10320	41421.2
10640	42846.7
13400	55303.2
14700	61259.6
15600	65412.7
21500	93147.4



3.3 ¿Sería viable?

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

Con base a lo resaltado por la gráfica, creemos que el algoritmo Insertion Sort no sería viable para aplicarlo en juegos ya que para estos necesitamos velocidades de renderización de los fotogramas muy altas y este código al crecer cuadráticamente en el tiempo nos dejaría con largas esperas tratándose de millones de elementos en n.

3.4 ¿Por qué?

El logaritmo que aparece en el algoritmo de Merge Sort se debe a que éste divide el array en sub-arrays y estos, si siguen siendo muy grandes, se dividen a su vez en sub-arrays. En otras palabras, el problema principal se divide en problemas alternos y estos de igual manera hasta que el problema sea resuelto y las soluciones alternas conformen la solución principal.

3.5¿Cómo?

Para arreglos grandes, Insertion Sort sería mas eficiente si el arreglo ya estuviese clasificado u ordenado, y esto se debe a que el algoritmo funciona más eficientemente de cualquier forma en arreglos pequeños.

4) Simulacro de Parcial

4.1: *C*
4.2: *B*
4.3: *B*
4.4: *A*
4.5:
  4.5.1: *D*
  4.5.2: *A*
4.6: *Se tardará 100000 segundos*
4.7: *Todas son verdaderas*
4.8: *A*
4.9: *A*
4.10: *C*
4.11: *C*
4.12: *B*
4.13: *C*
4.14: *A*

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