PARTE 3

Estudiante 1
Nombre: Juan Sebastian Pardo
Correo Uniandes: j.pardor
Codigo: 201923794
Especificaciones del computador:

- Modelo: MacBookAir 13”
- Memoria RAM: 4gb 1600MHz
- Procesador: 1.6Hz Dual-Core Intel Corei5

Estudiante 2
Nombre: Daniela ricaurte
Correo Uniandes: d.ricaurte
Codigo: 201822966
Especificaciones del computador:

- Modelo: MacBookAir 13”
- Memoria RAM: 4gb
- Procesador: 1,6 GHz Dual-Core Intel Core i5

Tabla 1: Tabla de Complejidad Temporal Teórica en el peor caso para cada algoritmo aplicado en una lista de N datos en representación Arreglo.

	Insertion Sort	Shell Sort	Merge Sort	Quick Sort
Complejidad O(…) en el peor caso para una muestra de N datos	$O(N^2)$	$O(\frac{N^3}{2})$	$O(N log\ N)$	$O(N^2)$

Tabla 2: Tabla de rendimiento para los distintos algoritmos de ordenamiento en Arreglo dinamico.

Tabla 2.1 Computador estudiante 1:

Tamaño de la muestra	Insertion Sort (tiempos en mseg)	Shell Sort (tiempos en mseg)	Merge Sort (tiempos en mseg)	Quick Sort (tiempos en mseg)
1000	55	8	Ver **NOTA 1**	11

2000	263	19		14
3000	279	25		29
4000	423	49		35
8000	3298	85		55
16000	12953	111		103
32000	747164	243		180
64000	t>600000	436		279
128000	t>600000	1321		866
256000	t>600000	2229		2453
512000	t>600000	4388		7423

Tabla 2.2 Computador estudiante 2:

Tamaño de la muestra	Insertion Sort (tiempos en mseg)	Shell Sort (tiempos en mseg)	Merge Sort (tiempos en mseg)	Quick Sort (tiempos en mseg)
1000	39	11	Ver **NOTA 1**	6
2000	279	15		17
3000	264	19		27
4000	429	46		26
8000	3107	81		49
16000	12912	106		97
32000	74719	237		195
64000	t>600000	415		286
128000	t>600000	1293		847
256000	t>600000	2238		2307
512000	t>600000	4375		7253

Tabla 3. Tabla de comparación de algoritmos de ordenamiento en Arreglo dinamico
Tabla 3.1 Computador estudiante 1:

	Alg. Más eficiente en tiempo	2o Alg. Más eficiente	3er. Alg. Más eficiente	Alg. menos eficiente en tiempo
Algoritmo	Merge sort	Quick sort	Shell sort	Insertiont sort

Tabla 3.2 Computador estudiante 2:

	Alg. Más eficiente en tiempo	2o Alg. Más eficiente	3er. Alg. Más eficiente	Alg. menos eficiente en tiempo
Algoritmo	Merge sort	Quick sort	Shell sort	Insertiont sort

Tabla 4. Tabla de rendimiento para los distintos algoritmos de ordenamiento en Lista Enlazada
Tabla 4.1 Computador estudiante 1:

Tamaño de la muestra	Insertion Sort (tiempos en mseg)	Shell Sort (tiempos en mseg)	Merge Sort (tiempos en mseg)	Quick Sort (tiempos en mseg)
1000	19095	1026	48	686
2000	230177	6406	406	4570
3000	t>600000	22008	733	17530
4000	t>600000	58805	1712	35634
8000	t>600000	407401	3610	174637
16000	t>600000	t>600000	2087	t>600000
32000	t>600000	t>600000	153299	t>600000
64000	t>600000	t>600000	t>600000	t>600000
128000	t>600000	t>600000	t>600000	t>600000

256000	t>600000	t>600000	t>600000	t>600000
512000	t>600000	t>600000	t>600000	t>600000

Tabla 4.2 Computador estudiante 2:

Tamaño de la muestra	Insertion Sort (tiempos en mseg)	Shell Sort (tiempos en mseg)	Merge Sort (tiempos en mseg)	Quick Sort (tiempos en mseg)
1000	21636	923	48	831
2000	230177	6406	423	4525
3000	t>600000	22008	792	17291
4000	t>600000	58805	1785	35512
8000	t>600000	407401	3674	174597
16000	t>600000	t>600000	2066	t>600000
32000	t>600000	t>600000	153393	t>600000
64000	t>600000	t>600000	t>600000	t>600000
128000	t>600000	t>600000	t>600000	t>600000
256000	t>600000	t>600000	t>600000	t>600000
512000	t>600000	t>600000	t>600000	t>600000

Tabla 5. Tabla de comparación de algoritmos de ordenamiento en Lista Enlazada.
Tabla 5.1 Computador estudiante 1:

	Alg. Más eficiente en tiempo	2o Alg. Más eficiente	3er. Alg. Más eficiente	Alg. menos eficiente en tiempo
Algoritmo	Merge Sort	Quick Sort	Shell Sort	Insertion Sort

Tabla 5.2 Computador estudiante 2:

	Alg. Más eficiente en tiempo	2o Alg. Más eficiente	3er. Alg. Más eficiente	Alg. menos eficiente en tiempo

Algoritmo	Merge Sort	Quick Sort	Shell Sort	Insertion Sort

Tabla 6. Tabla de comparación de cada algoritmo de ordenamiento de acuerdo a la Estructura de Datos utilizada (Arreglo y Lista Enlazada).

	Estructura de Datos Más eficiente en tiempo	Estructura de Datos Menos eficiente en tiempo
Insertion sort	Arreglo	Lista enlazada
Shell sort	Arreglo	Lista enlazada
Merge sort	Arreglo	Lista enlazada
Quick sort	Lista enlazada	Arreglo

Pregunta 1: A partir de la tabla anterior, en el caso general de ordenamiento ¿Cuál Estructura de Datos es mejor utilizar si solo se tiene en cuenta los tiempos de ejecución de los algoritmos? La Estructura de datos más eficiente en ejecución de tiempo es el arreglo dinámico, debido a que los tiempos de ordenamiento de todos los algoritmos son todos menores en tiempo de ejecución a los algoritmos de ordenamiento en las listas enlazadas.

NOTA 1

Tuvimos problemas con la implementación del método de ordenamiento Merge sort en los arreglos dinámicos, por esta razón no hay nada en ese campo. Hicimos todo lo posible en nuestro poder para arreglarlo pero fue imposible, estaba fuera de nuestras capacidades.