OBSERVACIONES DE LA PRÁCTICA

Maria Gabriela Herrera Rojas 202510147
Ileanne Valderrama Ocampo 202514355
Juan Pablo Peñaranda Cely 202422285

Ambientes de pruebas

	Computador 1 – Estudiante 1
Procesador	Intel(R) Core(TM) i5-10310U CPU @ 1.70GHz (2.21 GHz)
Memoria RAM (GB)	16.0 GB
Sistema Operativo	64-bit operating system, x64-based processor

Tabla 1. Especificaciones del computador para ejecutar las pruebas de rendimiento.

Para realizar las siguientes pruebas, es importante que las ejecuten utilizando el archivo de datos **large**, ya que es con este conjunto donde realmente se puede observar el comportamiento y la eficiencia de los algoritmos en contextos más exigentes.

Ordenamientos Iterativos

Resultados para ordenamientos iterativos con Array List

Porcentaje de la muestra	**Insertion Sort (Array List) [ms]**	**Selection Sort (Array List) [ms]**	**Shell Sort (Array List) [ms]**
0.50%	6106.496	8425.010	101.173
5.00%	0.022	0.086	0.032
10.00%	0.171	0.137	0.100
20.00%	0.314	0.680	0.107
30.00%	0.662	0.465	0.408
50.00%	0.722	2.422	0.974
80.00%	2.719	8.032	1.907
100.00%	3.368	9.640	0.571

*Tabla 2. **Resultados en computador 1 para ordenamientos iterativos con Array List.***

Resultados para ordenamientos iterativos con Linked List

Porcentaje de la muestra	Insertion Sort (Linked List)	Selection Sort (Linked List)	Shell Sort (Linked List)
0.50%	22.153	46.466	2.892
5.00%	0.034	0.043	0.104
10.00%	0.071	0.008	0.195
20.00%	0.327	0.271	0.221
30.00%	0.352	1.007	0.383
50.00%	0.938	1.393	2.933
80.00%	2.065	3.551	1.049
100.00%	3.449	2.442	1.787

Tabla 3. ***Resultados en computador 1 para ordenamientos iterativos con Single Linked List.***

Ordenamientos Recursivos

Resultados para ordenamientos recursivos con Array List

Porcentaje de la muestra	Merge Sort (Array List) [ms]	Quick Sort (Array List) [ms]
0.50%	80.12	165.222
5.00%	0.076	0.043
10.00%	0.087	0.152
20.00%	0.177	0.212
30.00%	0.260	0.295
50.00%	0.671	0.508
80.00%	0.893	1.165
100.00%	1.036	0.827

Tabla 4. ***Resultados en computador 1 para ordenamientos recursivos con Array List.***

Resultados para ordenamientos recursivos con Linked List

Porcentaje de la muestra	Merge Sort (Linked List)	Quick Sort (Linked List)
0.50%	5.588	5.114
5.00%	0.033	0.113
10.00%	0.078	0.151
20.00%	0.242	0.522
30.00%	0.271	1.338
50.00%	1.065	1.509
80.00%	1.564	1.106
100.00%	1.185	2.821

Tabla 5. ***Resultados en computador 1 para ordenamientos recursivos con Single Linked List.***

Una vez alla llenado l

Comparación de tiempo mejores algoritmos de ordenamiento

Complete esta sección únicamente cuando se le indique en las instrucciones. En este punto, deberá comparar el tiempo de ejecución entre el mejor algoritmo de ordenamiento recursivo y el mejor algoritmo de ordenamiento iterativo, los cuales debió haber identificado en el punto anterior.

Por favor, registre las mediciones en el espacio correspondiente de la tabla, especificando claramente:

- El nombre de cada algoritmo utilizado
- El tipo de estructura de datos empleada **(*ArrayList* o *SingleLinkedList*)**

Asegúrese de completar todos los campos solicitados en la tabla e incluir los nombres según corresponda, no es necesario que vuelva a correr las pruebas esta información la puede sacar de las tablas anteriores.

Computador 1

Tipo de estructura escogida: Array List

Algoritmo recursivo: Merge Sort

Algoritmo iterativo: Shell sort

Porcentaje de la muestra	Algoritmo recursivo	Algoritmo iterativo
0.50%	80.12	101.173
5.00%	0.076	0.032
10.00%	0.087	0.100
20.00%	0.177	0.107
30.00%	0.260	0.408
50.00%	0.671	0.974
80.00%	0.893	1.907
100.00%	1.036	0.571

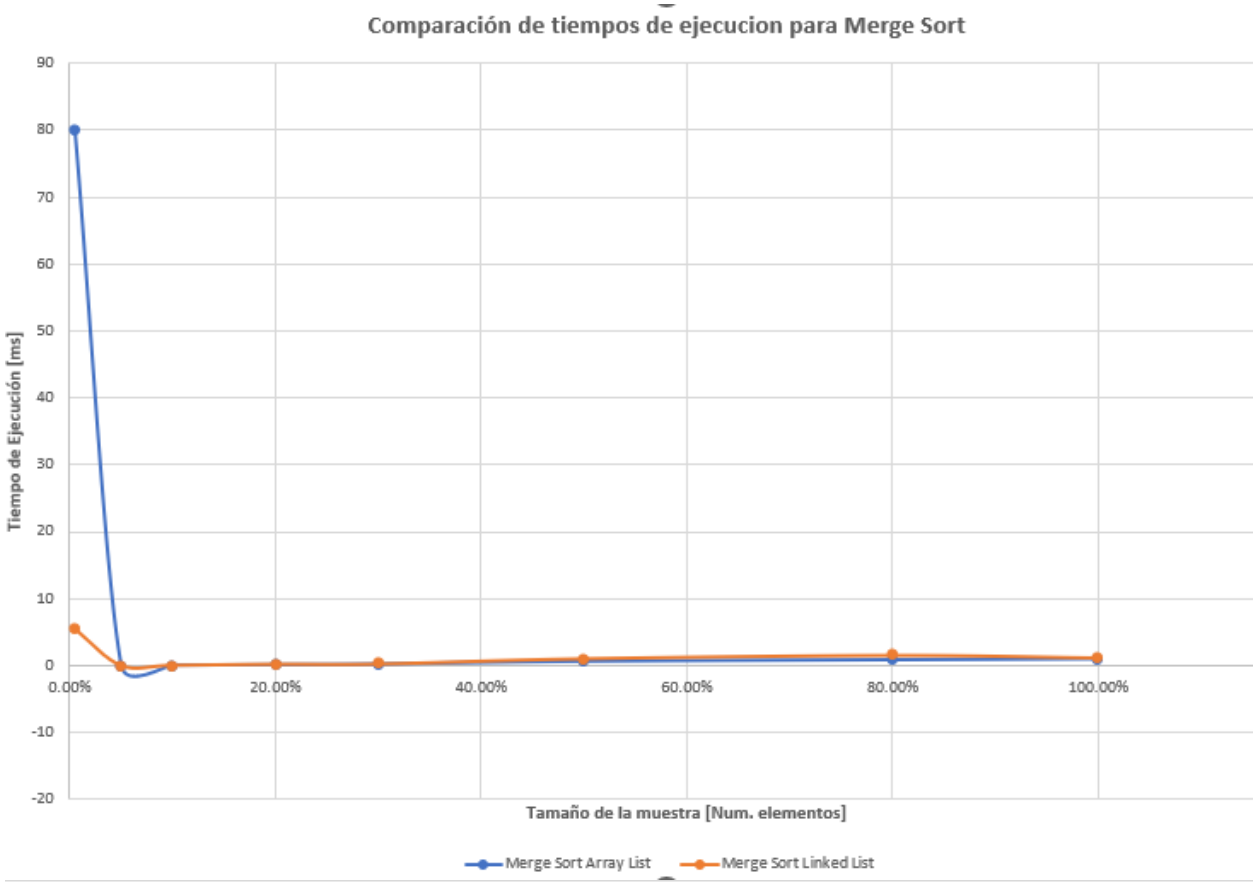
Tabla 6. ***Resultados en computador 1 mejores algoritmos***

Preguntas de análisis parte 1

1. **¿Cómo varía el comportamiento de cada algoritmo con respecto al tamaño de los datos?**

   Para responder, agregue líneas de tendencia en las gráficas de las pestañas 04 a 08**.** Observe si el crecimiento del tiempo de ejecución es lineal, cuadrático o de otro tipo, y relacione ese patrón con la complejidad teórica esperada de cada algoritmo.

   RTA: Cada algoritmo cada vez que se le va aumentando el tamaño, se le va incrementando su tiempo de ejecución, a excepción de los algoritmos iterativos como es el caso de Shell sort.

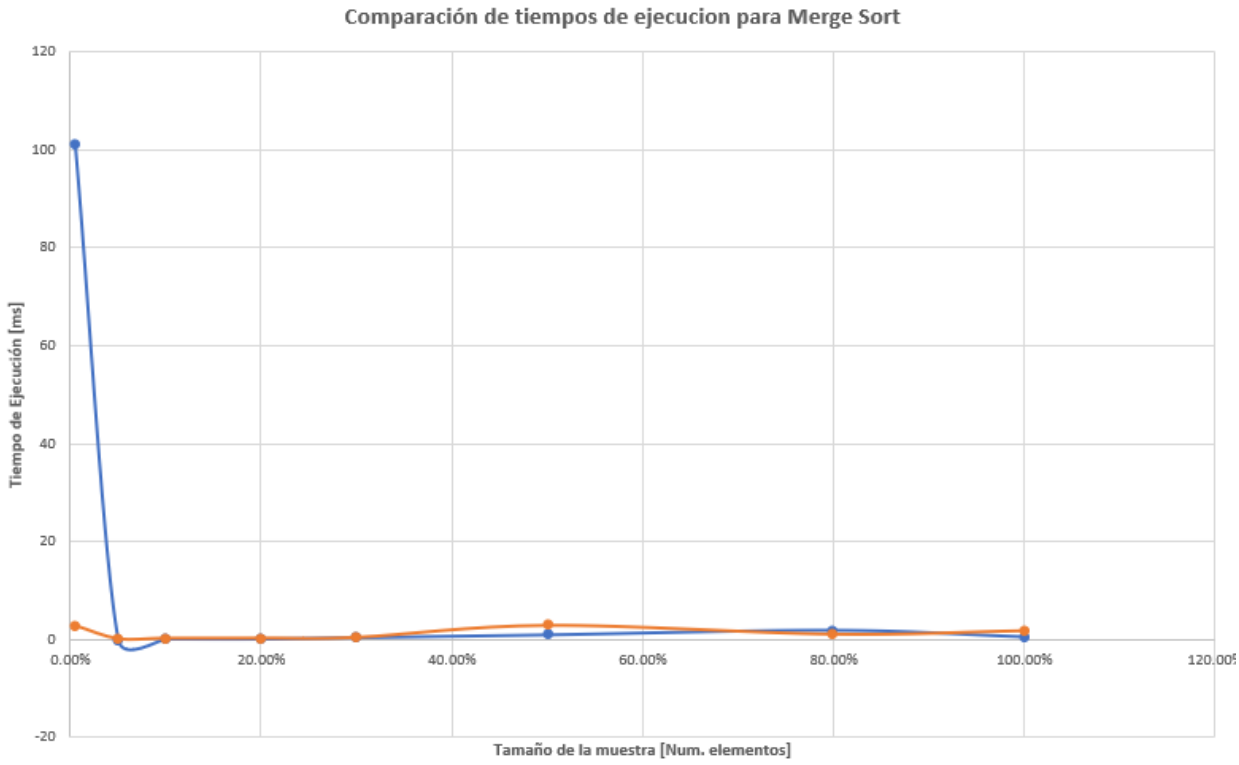

Comparación de tiempos de ejecucion para Merge Sort
Tiempo de Ejecución [ms]
90
80
70
60
50
40
30
20
10
0
-10
-20
0.00%
20.00%
40.00%
60.00%
80.00%
100.00%
Tamaño de la muestra [Num. elementos]
Merge Sort Array List
Merge Sort Linked List


Comparación de tiempos de ejecucion para Insertion Sort
Tiempo de Ejecución [ms]
7000
6000
5000
4000
3000
2000
1000
0
-1000
0.00%
20.00%
40.00%
60.00%
80.00%
100.00%
120.00%
Tamaño de la muestra [Num. elementos]
Insertion Sort Array List
Insertion Sort Linked List

2. **¿Qué diferencias observas en el rendimiento de un mismo algoritmo al utilizar ArrayList frente a SingleLinkedList?**

   Analice esta comparación en las pestañas 04 a 08, donde se muestran ambos casos. Justifique su respuesta con base en las líneas de tendencia.

   RTA: Pues aunque en este caso no se puede comparar ya que debido a que el archivo para singled por ser muy pesado se demoraba muchísimo en cargar, (me toco hacer un break con 500 libros para singled). Por tanto no es muy bueno comparar ya que no se encuentran bajos las mismas condiciones, sin embargo con esos valores, se nota bastante la diferencia que en singled el tiempo de ejecución es mayor ya que con solo 500 libros toma 46.466, sin embargo tanto en single como en array, el Shell sort es el algoritmo con menos tiempo de ejecución.

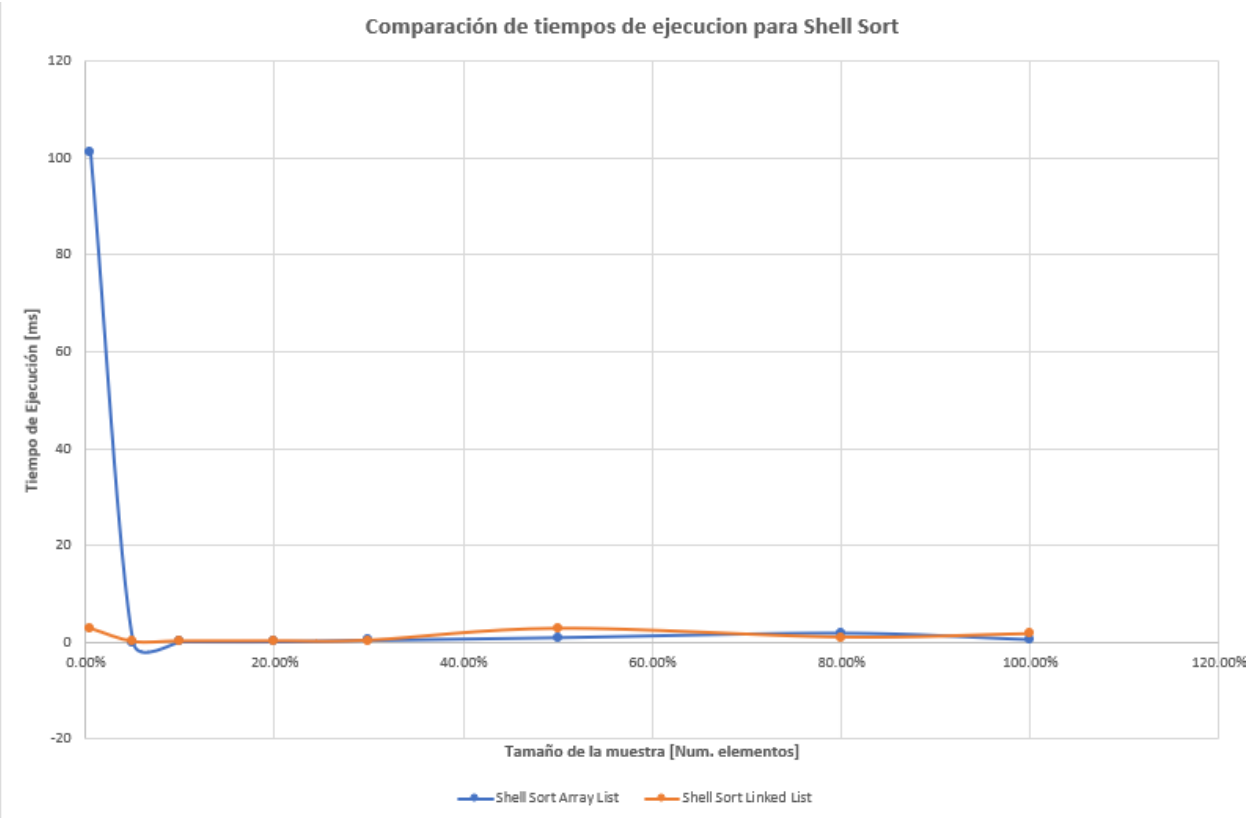


3. **¿Qué algoritmo iterativo mostró el mejor comportamiento general en términos de tiempo y cantidad de datos en ambas estructuras de datos?**

   Justifique su elección comparando las gráficas (04 Insertion Sort), (05 Selection Sort) y (06 Shell Sort) y apoyándose en las gráficas de las pestañas 2 y 3.

RTA: Shell sort en Array list claramente, ya que en array list se hizo la carga completa de los 10000 libros, y aun así obtuvo el menor de los tiempos de ejecución, aunque en singled este también obtuvo el menor tiempo de ejecución, la carga de libros o datos que

manejo fue muchísimo menor. Por tanto, el mejor algoritmo iterativo fue Shell sort en ambas estructuras de datos.



4. **¿Qué algoritmo recursivo presentó el mejor desempeño general según los tiempos registrados y la forma de la línea de tendencia?**

   Justifique su respuesta basándose en las pestañas 07 (Merge Sort) y 08 (Quick Sort) y apoyándose en las gráficas de las pestañas 2 y 3.

   RTA: Teniendo en cuenta las líneas de tendencia, el mejor algoritmo recursivo fue Quick sort ya que en ambas estructuras de datos, se mantuvo muy constante.


Comparación de tiempos de ejecucion para Quick Sort
Tiempo de Ejecución [ms]
180
160
140
120
100
80
60
40
20
0
-20
-40
0.00%
20.00%
40.00%
60.00%
80.00%
100.00%
120.00%
Tamaño de la muestra [Num. elementos]
Quick Sort Array List
Quick Sort Linked List


Comparación de tiempos de ejecucion para Merge Sort
Tiempo de Ejecución [ms]
90
80
70
60
50
40
30
20
10
0
-10
-20
0.00%
20.00%
40.00%
60.00%
80.00%
100.00%
120.00%
Tamaño de la muestra [Num. elementos]
Merge Sort Array List
Merge Sort Linked List

5. **Con base en su análisis visual y los datos recolectados, ¿cuáles algoritmos seleccionan como los "mejores" en cada categoría (iterativo y recursivo)?**

   Indique también la estructura de datos en la que se desempeñó mejor cada uno.

   RTA: En iterativo selecciono al mejor algoritmo, Shell sort en cada estructura de datos fue disminuyendo su timepoo. Y en recursivo, selecciono a Quick sort.

Preguntas de análisis parte 2

1. **¿Cuál de los dos algoritmos seleccionados como "mejores" muestra un comportamiento más eficiente a medida que crece el tamaño de la muestra?**

   Para responder, agregue líneas de tendencia en la gráfica de la pestaña 09-Bests, identifique cuál algoritmo crece más lentamente y relacione ese patrón con su complejidad teórica. Use esta comparación para justificar cuál es más escalable.

   RTA:

2. **¿Qué estructura de datos resultó más favorable para cada algoritmo seleccionado como mejor?**

   Analice si la ventaja se mantiene constante en todos los tamaños o solo en ciertos rangos.

   RTA: Yo creo que la estructura de datos mas eficiente fue claramente array list, ya que esta permitió la carga completa de datos y sus tiempos de ejecución fueron bastante buenos en comparación a los de singled, que no tuvo toda la carga de datos.

3. **¿Qué ventajas prácticas podría tener implementar el algoritmo ganador en aplicaciones reales donde se manejan grandes volúmenes de datos?**

   RTA: Mayor eficiencia y rapidez, al momento de la carga de datos.

4. **Si se presentara una lista parcialmente ordenada, ¿cambiaría su elección de algoritmo? Explique por qué, con base en el comportamiento observado.**

   RTA: De algoritmo creo que si cambiaria a Merge sort, ya que seria mas fácil y mas rápido el tema de las comparaciones, y si la lista se encuentra prácticamente ordenada no se tendría que hacer tantos cambios entre posiciones etc..

Ambientes de pruebas

	Computador 2
Procesador	Intel(R) Core(TM) i3-10110U CPU @ 2.10GHz (2.59 GHz)
Memoria RAM (GB)	16 GB
Sistema Operativo	Sistema operativo de 64 bits, procesador x64

Tabla 2. Especificaciones del computador para ejecutar las pruebas de rendimiento.

Para realizar las siguientes pruebas, es importante que las ejecuten utilizando el archivo de datos **large**, ya que es con este conjunto donde realmente se puede observar el comportamiento y la eficiencia de los algoritmos en contextos más exigentes.

Ordenamientos Iterativos

Resultados para ordenamientos iterativos con Array List

Porcentaje de la muestra	Insertion Sort (Array List) [ms]	Selection Sort (Array List) [ms]	Shell Sort (Array List) [ms]
0.50%	7642.2	5617.14	97.24
5.00%	1.2	0.09	0.02
10.00%	1.2	1.3	0.08
20.00%	1.451	1.72	1.7
30.00%	1.34	2.0	3.1
50.00%	1.76	3.7	10.0000
80.00%	1.95	7.6235	12.344
100.00%	2.6	9.8231	13.002

*Tabla 2. **Resultados en computador 1 para ordenamientos iterativos con Array List.***

Resultados para ordenamientos iterativos con Linked List

Porcentaje de la muestra	Insertion Sort (Linked List)	Selection Sort (Linked List)	Shell Sort (Linked List)
0.50%	25.2	52.138	3.4987
5.00%	0.0297	0.0538	0.0633
10.00%	0.045	0.0105	0.835
20.00%	0.0523	0.149	0.9342
30.00%	0.136	0.3493	1.29
50.00%	1.384	2.739	2.0
80.00%	1.9338	4.99383	1.789

100.00%	2.0349	6.067	2.01

Tabla 3. ***Resultados en computador 1 para ordenamientos iterativos con Single Linked List.***

Ordenamientos Recursivos

Resultados para ordenamientos recursivos con Array List

Porcentaje de la muestra	Merge Sort (Array List) [ms]	Quick Sort (Array List) [ms]
0.50%	100.1	200.92
5.00%	0.08	0.0522
10.00%	0.1	0.203
20.00%	0.293	0.283
30.00%	0.448	0.293
50.00%	0.7993	0.681
80.00%	1.0939	1.1382
100.00%	1.5	0.11

Tabla 4. ***Resultados en computador 1 para ordenamientos recursivos con Array List.***

Resultados para ordenamientos recursivos con Linked List

Porcentaje de la muestra	Merge Sort (Linked List)	Quick Sort (Linked List)
0.50%	8.389	0.04
5.00%	0.05	0.04
10.00%	0.079	0.0529
20.00%	0.05	0.07
30.00%	0.092	0.09
50.00%	0.1237	1.394
80.00%	1.04	3.038
100.00%	3.049	5.3483

Tabla 5. ***Resultados en computador 1 para ordenamientos recursivos con Single Linked List.***

Comparación de tiempo mejores algoritmos de ordenamiento

Complete esta sección únicamente cuando se le indique en las instrucciones. En este punto, deberá comparar el tiempo de ejecución entre el mejor algoritmo de ordenamiento recursivo y el mejor algoritmo de ordenamiento iterativo, los cuales debió haber identificado en el punto anterior.

Por favor, registre las mediciones en el espacio correspondiente de la tabla, especificando claramente:

- El nombre de cada algoritmo utilizado
- El tipo de estructura de datos empleada (***ArrayList*** **o** ***SingleLinkedList***)

Asegúrese de completar todos los campos solicitados en la tabla e incluir los nombres según corresponda, no es necesario que vuelva a correr las pruebas esta información la puede sacar de las tablas anteriores.

Computador 2

Tipo de estructura escogida: Array List

Algoritmo recursivo: Merge Sort

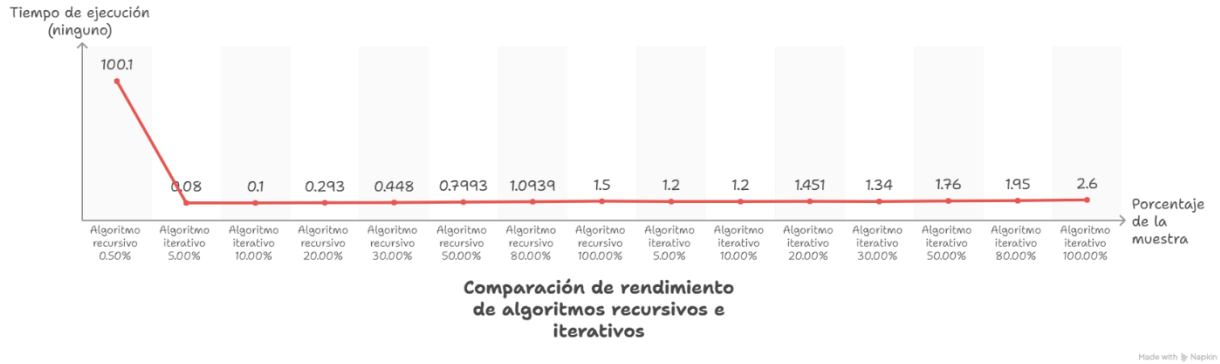
Algoritmo iterativo: Insertion sort

Porcentaje de la muestra	Algoritmo recursivo	Algoritmo iterativo
0.50%	100.1	7642.2
5.00%	0.08	1.2
10.00%	0.1	1.2
20.00%	0.293	1.451
30.00%	0.448	1.34
50.00%	0.7993	1.76
80.00%	1.0939	1.95
100.00%	1.5	2.6

Tabla 6. ***Resultados en computador 1 mejores algoritmo***

Preguntas de análisis parte 1

Grafica:



1. **¿Cómo varía el comportamiento de cada algoritmo con respecto al tamaño de los datos?**

   Para responder, agregue líneas de tendencia en las gráficas de las pestañas 04 a 08**.** Observe si el crecimiento del tiempo de ejecución es lineal, cuadrático o de otro tipo, y relacione ese patrón con la complejidad teórica esperada de cada algoritmo.

RTA: En insertion sort se ve que aumenta casi que el doble del tiempo desde el 30% al 100%, mientras que en merge sort aumenta un poco más del triple el tiempo, lo que hace a ambas, lineales.

2. **¿Qué diferencias observas en el rendimiento de un mismo algoritmo al utilizar ArrayList frente a SingleLinkedList?**

Analice esta comparación en las pestañas 04 a 08, donde se muestran ambos casos. Justifique su respuesta con base en las líneas de tendencia.

RTA: Array list funciona más para buscar elementos por indexación, aunque single linked list funciona mejor al momento de hacer uso de funciones como merge sort gracias a su eficiencia para el movimiento entre nodos y cambio de su posición.

3. **¿Qué algoritmo iterativo mostró el mejor comportamiento general en términos de tiempo y cantidad de datos en ambas estructuras de datos?**

Justifique su elección comparando las gráficas (04 Insertion Sort), (05 Selection Sort) y (06 Shell Sort) y apoyándose en las gráficas de las pestañas 2 y 3.

RTA: Desde mi punto de vista la que tiene mejores tiempos y comportamiento es la de insertion sort.

4. **¿Qué algoritmo recursivo presentó el mejor desempeño general según los tiempos registrados y la forma de la línea de tendencia?**

Justifique su respuesta basándose en las pestañas 07 (Merge Sort) y 08 (Quick Sort) y apoyándose en las gráficas de las pestañas 2 y 3.

RTA: Merge sort en la tabla tiene un mejor comportamiento, por recursividad y tiempos

.

5. **Con base en su análisis visual y los datos recolectados, ¿cuáles algoritmos seleccionan como los "mejores" en cada categoría (iterativo y recursivo)?**

Indique también la estructura de datos en la que se desempeñó mejor cada uno.

RTA: El mejor de los iterativos es el insertion sort y de los recursivos el merge sort.

Preguntas de análisis parte 2

1. **¿Cuál de los dos algoritmos seleccionados como "mejores" muestra un comportamiento más eficiente a medida que crece el tamaño de la muestra?**

Para responder, agregue líneas de tendencia en la gráfica de la pestaña 09-Bests, identifique cuál algoritmo crece más lentamente y relacione ese patrón con su complejidad teórica. Use esta comparación para justificar cuál es más escalable.

RTA: Merge sort es más eficiente y mejor que el insertion sort, lo cual se evidencia en las tablas.

2. **¿Qué estructura de datos resultó más favorable para cada algoritmo seleccionado como mejor?**

Analice si la ventaja se mantiene constante en todos los tamaños o solo en ciertos rangos.

RTA:

3. **¿Qué ventajas prácticas podría tener implementar el algoritmo ganador en aplicaciones reales donde se manejan grandes volúmenes de datos?**

RTA: Merge sort podría acomodarse mejor a single linked list para usar un manejo más apropiado de los recursos.

4. **Si se presentara una lista parcialmente ordenada, ¿cambiaría su elección de algoritmo? Explique por qué, con base en el comportamiento observado.**

RTA:

En listas parcialmente ordenadas, insertion sort puede ser mas eficiiente porque aprovecha el orden previo y se acerca a

Juan Pablo? Qué haces, es una por computador?