LABORATORIO NO. 4: ORDENAMIENTOS ITERATIVOS & RECURSIVOS

Objetivos

Comprender el funcionamiento y la implementación de los algoritmos de ordenamiento básicos.

Al finalizar este laboratorio el estudiante estará en capacidad de:

- Comprender la complejidad (temporal y espacial) de los algoritmos de ordenamiento iterativos y recursivos.
- Validar los tiempos de ejecución de los ordenamientos desarrollando pruebas funcionales.
- Comprender la diferencia entre ordenamientos iterativos y recursivos.
- Integrar código de monitoreo (toma de datos) dentro de lo previamente implementado.
- Seguir utilizando adecuadamente el ambiente de trabajo (VS Code, GIT y GitHub)

Fecha Límite de Entrega

Miércoles 15 de septiembre antes de la media noche (11:59 p.m.).

Preparación del Laboratorio

- Revisar las implementaciones del **ADT List** y las Estructuras de Datos Arreglo **(ARRAYLIST)** y Lista sencillamente Encadenada **(LINKED_LIST).** Estas implementaciones se encuentran en la carpeta **DISCLib** de sus repositorios de los laboratorios anteriores.
- Revisar las implementaciones de los algoritmos de ordenamiento iterativos y recursivos (Selection, Shell, Quick y Merge Sorts) disponibles en la carpeta **DISCLib/Algorithms/Sorting** en los repositorios de los laboratorios.
- El repositorio oficial del laboratorio contiene los archivos esqueleto **observaciones-lab4.docx** y **TablasDatos-Lab4.xlsx** que les ayudarán a resolver las preguntas de observación, registrar los tiempos de ejecución y graficar las líneas de tendencia de los experimentos para el Reto.
- En algunos casos experimentales puede que Python y el IDE declaren que se alcanzó el límite de recursión con un mensaje **"`RecursionError: maximum recursion depth exceeded in comparison`"**, en este caso se recomienda actualizar en el **view.py** este límite con las siguientes agregando las siguientes líneas de código:

```
import sys
…
default_limit = 1000
sys.setrecursionlimit(default_limit*10)
```

Trabajo Propuesto

PASO 1: Copiar el ejemplo en su organización

Copie/Haga **Fork** del repositorio del laboratorio en su organización con el procedimiento aprendido en las prácticas anteriores.

El repositorio del proyecto base que utiliza este laboratorio es el siguiente:

- https://github.com/ISIS1225DEVS/ISIS1225-SampleSorts.git

Antes de clonar el repositorio en su computador diríjase a su organización (Ej.: ***EDA2021-2-SEC02-G01*** para el primer grupo de la sección 2 del curso) y cambie el nombre del repositorio de acuerdo con el esquema **LabSorts-S<<XX>>-G<<YY>>** donde **XX** es el número de la semana de la práctica y donde **YY** es el número del grupo de trabajo. (Ej.: **LabSorts-S04-G01** para este **cuarto laboratorio** hecho por el **grupo 1** de la **sección 2).**

Recuerde que **NO necesita** agregar la sección o el semestre en este nombre porque ya está identificado en su organización.

PASO 2: Descargar el ejemplo

Después de renombrar el proyecto dentro de su organización ya puede clonar el proyecto. Descargue el código en su máquina local siguiendo lo aprendido en las practicas anteriores.

Recuerde modificar el **README** del repositorio para incluir los nombres de los integrantes del grupo.

PASO 3: Estudiar y Modificar el ejemplo

El proyecto **SampleSort** busca familiarizarlos con las funciones de ordenamiento y una forma de probar su desempeño en aplicaciones MVC.

Antes de iniciar a explorar y modificar el ejemplo, recuerde descargar los datos de trabajo ***Goodreads*** disponibles en el portal oficial del curso en BrightSpace. Descargue el **Zip,** descomprímalo y guarde los archivos CSV en la carpeta ***/Data/GoodReads/** de su copia local de código.

Diríjase al archivo **`controller.py`** y cambie los archivos que está cargando el programa, para ello borre el sufijo **"-small"** de las funciones **`loadBooks()`** y **`loadBooksTags().`**

Al terminar estas modificaciones, diríjase al archivo **`view.py`** para ejecutar el menú, al hacerlo podrá ver el siguiente mensaje en la consola.

```
Bienvenido
1- Cargar información en el catálogo
2- Consultar los Top x libros por promedio
3- Consultar los libros de un autor
4- Libros por género
5 - Ordenar los libros por rating
0- Salir
Seleccione una opción para continuar
```

Cargue los datos ejecutando la **opción 1** y revise el número de datos cargados en el catálogo como se muestra a continuación.

```
Seleccione una opción para continuar
1
Cargando información de los archivos ....
Libros cargados: 10000
Autores cargados: 5833
Géneros cargados: 34252
Asociación de Géneros a Libros cargados: 999912
Bienvenido
1- Cargar información en el catálogo
2- Consultar los Top x libros por promedio
3- Consultar los libros de un autor
4- Libros por género
5 - Ordenar los libros por rating
0- Salir
Seleccione una opción para continuar
```

después ejecute la **opción 5,** aquí el programa le pedirá un tamaño de muestra para ejecutar un ordenamiento, en el ejemplo elegiremos una submuestra de **5000 libros**. Al terminar la ejecución podrá ver un mensaje como el siguiente.

```
Seleccione una opción para continuar
5
Indique tamaño de la muestra: 5000
Para la muestra de 5000  elementos, el tiempo (mseg) es:  40281.25
Bienvenido
1- Cargar información en el catálogo
2- Consultar los Top x libros por promedio
3- Consultar los libros de un autor
4- Libros por género
5 - Ordenar los libros por rating
0- Salir
Seleccione una opción para continuar
```

Ahora, abra el archivo **model.py** y diríjase a las marcas de **TODO** para realizar una revisión del código y comprender cómo funciona el ordenamiento ejecutado en el paso anterior. Busque la función **sortBooks()** y modifique esta función para que, adicional al tiempo de ejecución, devuelva la lista ordenada de libros.

```python
def sortBooks(catalog, size):
    sub_list = lt.subList(catalog['books'], 1, size)
    sub_list = sub_list.copy()
    start_time = time.process_time()
    sorted_list = sa.sort(sub_list, compareratings)
    stop_time = time.process_time()
    elapsed_time_mseg = (stop_time - start_time)*1000
    return elapsed_time_mseg, sorted_list
```

A continuación, modifique en el archivo **view.py** agregando la función **printSortResults()** la cual modificará la variable **result** de la impresión por consola de la **opción 5** como se muestra en los siguientes bloques de código.

El código para agregar la función **printSortResults()** es:

```
def printSortResults(ord_books, sample=10):
    size = lt.size(ord_books)
    if size > sample:
        print("Los primeros ", sample, " libros ordenados son:")
        i=1
        while i <= sample:
            book = lt.getElement(ord_books,i)
            print('Titulo: ' + book['title'] + '  ISBN: ' +
                  book['isbn'] + ' Rating: ' + book['average_rating'])
            i+=1
```

La modificación en la **opción 5** del menú debe ser como se muestra a continuación, donde **result** en el **print()** pasa a ser **result[0]** y **printResults()** recibe como parámetro **result[1]**.

```
    elif int(inputs[0]) == 5:
        size = input("Indique tamaño de la muestra: ")
        result = controller.sortBooks(catalog, int(size))
        print("Para la muestra de", size, " elementos, el tiempo (mseg) es: ",
                                           str(result[0]))
        printSortResults(result[1])
```

Al terminar las modificaciones, ejecute el archivo **view.py,** cargue los archivos con la **opción 1** y vuelva a ejecutar la **opción 5** con una muestra **de 5000 libros**. Esto deberá mostrar un resultado similar al que se muestra a continuación.

```
Seleccione una opción para continuar
5
Indique tamaño de la muestra: 5000
Para la muestra de 5000  elementos, el tiempo (mseg) es:  38750.0
Los primeros  10  libros ordenados son:
Titulo: The Complete Calvin and Hobbes  ISBN: 740748475 Rating: 4.82
Titulo: The Complete Calvin and Hobbes  ISBN: 740748475 Rating: 4.82
Titulo: Words of Radiance (The Stormlight Archive, #2)  ISBN: 765326361 Rating: 4.77
Titulo: Harry Potter Boxed Set, Books 1-5 (Harry Potter, #1-5)  ISBN: 439682584 Rating: 4.77
Titulo: It's a Magical World: A Calvin and Hobbes Collection  ISBN: 836221362 Rating: 4.75
Titulo: Harry Potter Boxset (Harry Potter, #1-7)  ISBN: 545044251 Rating: 4.74
Titulo: Harry Potter Collection (Harry Potter, #1-6)  ISBN: 439827604 Rating: 4.73
Titulo: A Court of Mist and Fury (A Court of Thorns and Roses, #2)  ISBN:  Rating: 4.72
Titulo: The Holy Bible: English Standard Version  ISBN:  Rating: 4.66
Titulo: The Harry Potter Collection 1-4 (Harry Potter, #1-4)  ISBN: 439249546 Rating: 4.66
Titulo: The Essential Calvin and Hobbes: A Calvin and Hobbes Treasury  ISBN: 836218051 Rating: 4.65
Bienvenido
1- Cargar información en el catálogo
2- Consultar los Top x libros por promedio
3- Consultar los libros de un autor
4- Libros por género
5 - Ordenar los libros por rating
0- Salir
Seleccione una opción para continuar
```

Ahora diríjase al archivo **model.py,** y modifique la comparación de la función **compareratings()** de mayor **">"** a menor **"<"** como se muestra a continuación.

```
def compareratings(book1, book2):
    return (float(book1['average_rating']) < float(book2['average_rating']))
```

Reinicie la aplicación y vuelva a ejecutar la **opción 1** y la **opción 5** con una muestra de **5000 libros.** Esto deberá mostrar un resultado simular al siguiente.

```
Seleccione una opción para continuar
5
Indique tamaño de la muestra: 5000
Para la muestra de 5000  elementos, el tiempo (mseg) es:  37593.75
Los primeros  10  libros ordenados son:
Titulo: One Night at the Call Center  ISBN: 345498321 Rating: 2.47
Titulo: One Night at the Call Center  ISBN: 345498321 Rating: 2.47
Titulo: The Almost Moon  ISBN: 316677469 Rating: 2.67
Titulo: Four Blondes  ISBN: 080213825X Rating: 2.8
Titulo: Revenge Wears Prada: The Devil Returns (The Devil Wears Prada, #2)  ISBN: 1439136637 Rating: 2.84
Titulo: The Emperor's Children  ISBN: 030726419X Rating: 2.93
Titulo: Pygmy  ISBN: 385526342 Rating: 2.96
Titulo: The 3 Mistakes of My Life  ISBN:  Rating: 2.97
Titulo: Adultery  ISBN: 1101874082 Rating: 3.01
Titulo: Revolution 2020: Love, Corruption, Ambition  ISBN: 8129118807 Rating: 3.07
Titulo: The Jane Austen Book Club  ISBN: 452286530 Rating: 3.07
Bienvenido
1- Cargar información en el catálogo
2- Consultar los Top x libros por promedio
3- Consultar los libros de un autor
4- Libros por género
5 - Ordenar los libros por rating
0- Salir
Seleccione una opción para continuar
```

Note que ahora la **calificación (Rating)** cambia y la lista ordenada va de menor a mayor y no de mayor a menor como en la previa ejecución. **RECUERDE** esto al seguir con las instrucciones de este laboratorio.

PASO 6: Actualizar el repositorio en la rama principal

Confirme los cambios con los comandos **Commit** y **Push** en la rama ***main*** local y de GitHub con el comentario *"Entrega Ejemplo – laboratorio 4"* antes de la fecha límite de entrega de la parte 1.

Ordenamientos Iterativos y Recursivos

Después de familiarizarse con como ejecutar pruebas sobre los algoritmos de ordenamiento en **DISClib** y su uso en el **MVC** del curso diríjase a su repositorio del **Reto No. 1**.

Utilice TAD Lista más los ordenamientos iterativos y recursivos vistos en clase para organizar las obras de arte (artworks) utilizando la fecha de adquisición (AquiredDate) como criterio de comparación.

Esta funcionalidad es similar al **Requerimiento 2** del reto y sirve como un avance sobre el mismo.

PASO 1: Implementar Ordenamientos de obras de arte

Para completar la implementación de ***Requerimiento 2*** debe modificar los archivos **model.py** y **controller.py** e integrar los cambios en yo el archivo **view.py.** Para hacerlo exitosamente y completar las pruebas de los algoritmos de ordenamientos recuerde las siguientes indicaciones.

1) Descargue los archivos del MoMA para el **Reto No. 1** disponibles en la página oficial del curso en Brightspace y ubíquelos en la carpeta **Data** dentro del repositorio.
2) Modifique el código necesario en los archivos **model.py, controller.py,** y **view.py** para que en la **opción 1** del menú el usuario pueda seleccionar el tipo de representación de la lista **(ARRAY_LIST** o **LINKED_LIST)** en donde cargar el catálogo del museo.
3) Implemente el código de una función de ordenamiento (**cmpfunction**) que les permita comparar dos obras de arte teniendo en cuenta la fecha de adquisición de la obra (**DateAcquired**), útil para el ***Requerimiento 2.***

   La definición de esta función de ordenamiento es:

```
def cmpArtworkByDateAcquired(artwork1, artwork2):
    """
    Devuelve verdadero (True) si el 'DateAcquired' de artwork1 es menores que el de artwork2
    Args:
        artwork1: informacion de la primera obra que incluye su valor 'DateAcquired'
        artwork2: informacion de la segunda obra que incluye su valor 'DateAcquired'
    """
```

4) Implemente el código necesario que le permita al usuario elegir el tamaño de la muestra (sub-lista) y asegurarse que el tamaño de dicha muestra no exceda el tamaño de los datos cargados en memoria.

   TIP: En el archivo **DISCLib/ATD/List.py** esta implementada la función **sublist().**

5) Modifique el código necesario en los archivos **model.py, controller.py,** y **view.py** para que en la **opción 2** del menú el usuario pueda seleccionar el tipo de algoritmo de ordenamiento iterativo **(Insertion, Shell, Merge o Quick Sorts)** con el cuál ordenar el catálogo de las obras de arte por la fecha de adquisición.

   TIP: revise los archivos en DISC **DISCLib/ATD/Algorithms/Sorting/*** para familiarizarse con los algoritmos de ordenamiento.

6) Implementar el código de tal manera que pueda incluir la medición del tiempo de procesamiento del ordenamiento, para este fin revise la implementación realizada en el proyecto de ejemplo **SampleSort.**

Al finalizar esta implementación la **opción 3** del menú debe ejecutar sin problemas. Subsecuentemente, asegúrese de hacer **commit** y **push** de los cambios en el repositorio.

PASO 2: Preparación para las pruebas de rendimiento

Antes de iniciar las pruebas de esta práctica recomendamos tener en cuenta las siguientes instrucciones:

1) Cada estudiante debe ejecutar todas las pruebas de rendimiento propuestas en este aparte.
2) En el documento **Observaciones-lab4.docx** diligencia la Tabla 1 con la información de las máquinas de cómputo donde cada estudiante ejecutará las pruebas de los algoritmos de ordenamiento iterativo.

	Máquina 1	Máquina 2
Procesadores		
Memoria RAM (GB)		
Sistema Operativo		

Tabla 1. Especificaciones de las máquinas para ejecutar las pruebas de rendimiento.

Por ejemplo:

	Máquina 1	Máquina 2
Procesadores	Intel(R) Core(TM) i7-55000 CPU @2.40GHz 2.40GHZ	Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz 2.59 GHz
Memoria RAM (GB)	16.0 GBGB	32.0 GB
Sistema Operativo	Windows 10 Pro 64-bits	Windows 10 Pro 64-bits

3) Ejecute todas las pruebas de sobre los tiempos de ejecución que pueda en la misma máquina que reporto.
4) Cierre todas las aplicaciones que puedan afectar la toma de datos y sean **innecesarias** de su máquina, ej.: **Discord**, **OneDrive**, **Dropbox**, **Word** y en especial el navegador de internet (**Edge**, **Chrome**, **Firefox**, **Safari**).
5) Se recomienda que cada prueba se ejecute por lo menos **tres veces** para poder obtener un resultado que permita consolidar un resultado consistente de las pruebas. Ej. Para una muestra de 1000 elementos se debe ejecutar el Insertion Sort por lo menos 3 veces (ideal 5 veces) evitando errores e interrupciones dentro de la misma máquina, para luego promediar el resultado y registrarlo en las Tabla 2 y Tabla 3.
6) Cada uno de los tiempos debe estas registrado en **milisegundos (ms)** y con 2 cifras decimales redondeado hacia arriba desde la mitad. Ej.: 43494.4985 ms se aproxima a 43494.50 ms

PASO 3: Ejecutar pruebas de rendimiento

A continuación, cada uno de los estudiantes ejecutara las pruebas de rendimiento para cada uno de los algoritmos.

IMPORTANTE: El subconjunto de datos Small se utilizará como caso base para asegurarse que la implementación del código es adecuada

1) Inicie la aplicación y cargue los datos con la **Opción 1** en el catálogo de obras de arte (artwork) con una representación **`ARRAY_LIST.`**
2) Ejecute la **Opción 3,** seleccionando el primer ordenamiento de la Tabla 2 **(Insertion),** el tamaño de la muestra **(-small).**
3) Registre el tiempo de ejecución del algoritmo.
4) Repita los dos pasos anteriores por lo menos tres veces para calcular el promedio aritmético.
5) Después de obtener el tiempo promedio repita el procedimiento con la otra muestra **(10.00%)** y los demás algoritmos de ordenamiento **(Shell, Insertion, Quick y Merge)** hasta completar la tabla de datos o hasta donde lo permita sus recursos computacionales.
6) Reporte la Tabla 2 en el documento **`Observaciones-lab4.docx`**.

Porcentaje de la muestra [pct]	Tamaño de la muestra (ARRAYLIST)	Insertion Sort [ms]	Shell Sort [ms]	Quick Sort [ms]	Merge Sort [ms]
small					
10.00%					

Tabla 2. Comparación de tiempos de ejecución para los ordenamientos en la representación arreglo.

Después de completar la Tabla 2 cierre la aplicación y reinicie la aplicación y siga estas instrucciones:

1) Inicie la aplicación y cargue los datos con la **Opción 1** en el catálogo de obras de arte con una representación **`LINKED_LIST.`**
2) Ejecute la **Opción 3,** seleccionando el primer ordenamiento de la Tabla 2 **(Insertion),** el tamaño de la muestra **(-small).**
3) Registre el tiempo de ejecución del algoritmo.
4) Repita los dos pasos anteriores por lo menos tres veces para calcular el promedio aritmético.
5) Después de obtener el tiempo promedio repita el procedimiento con la otra muestra **(10.00%)** y los demás algoritmos de ordenamiento **(Shell, Insertion, Quick y Merge)** hasta completar la tabla de datos.
6) Reporte en la Tabla 3 en el documento **`Observaciones-lab4.docx`**.

Porcentaje de la muestra [pct]	Tamaño de la muestra (LINKED_LIST)	Insertion Sort [ms]	Shell Sort [ms]	Quick Sort [ms]	Merge Sort [ms]
small					
10.00%					

Tabla 3. Comparación de tiempos de ejecución para los ordenamientos en la representación lista enlazada.

PASO 4: Comparar algoritmos y representaciones

Después de tomar los datos, cada uno de los integrantes del grupo registrará los resultados en el documento de observaciones para comparar el comportamiento de los algoritmos y responderá unas preguntas de análisis.

Para ello cada uno de los estudiantes cumplirá con las siguientes instrucciones

1) Compare los resultados obtenidos con la complejidad teórica de cada algoritmo y con los resultados obtenidos de las pruebas en ambas maquinas respondiendo las siguientes preguntas:

    - ¿El comportamiento con relación al orden de crecimiento temporal de los algoritmos es acorde a lo enunciado teóricamente?
    - ¿Existe alguna diferencia entre los resultados obtenidos al ejecutar las pruebas en diferentes máquinas?
    - De existir diferencias, ¿a qué creen que se deben?

2) Registre las respuestas a las preguntas en el documento **`Observaciones-lab4.docx`** y guárdelo en formato PDF.
3) Tomando como base las tablas de datos complete la Tabla 4 en el documento **`Observaciones-lab4.docx`** e indique cual algoritmo es más eficiente en cual representación.

Algoritmo	Arreglo (ARRAYLIST)	Lista enlazada (LINKED_LIST)
Insertion Sort		
Shell Sort		
Merge Sort		
Quick Sort		

Tabla 4. Comparación de eficiencia de acuerdo con los algoritmos de ordenamientos y estructuras de datos utilizadas.

A partir de la Tabla 4 responda la siguiente pregunta en el documento **`Observaciones-lab4.docx`**:

- ¿Cuál Estructura de Datos (ARRAY_LIST o SINGLE_LINKED) funciona generalmente mejor si solo se tiene en cuenta los tiempos de ejecución de los algoritmos?
- Teniendo en cuenta las pruebas de tiempo de ejecución reportadas por los algoritmos de ordenamiento probados (iterativos y recursivos), proponga un listado de estos ordenarlos de menor a mayor teniendo en cuenta el tiempo de ejecución que toma ordenar las obras de arte.

4) Registre las respuestas a las preguntas en el documento **`Observaciones-lab4.docx`**

PASO 5: Documento de análisis de los ordenamientos.

Al completar las pruebas, y responder las preguntas verifique que todos los resultados se encuentren compilados en el archivo llamado **`Observaciones-lab4.docx`** dentro de la carpeta **`Docs`**.

El documento debe incluir lo siguiente:

- Nombres, apellidos, código de estudiante de los integrantes del grupo.
- Incluir los datos reportados en la Tabla 1, Tabla 2, Tabla 3 y Tabla 4.
- Respuesta a las preguntas de análisis.

PASO 6: Actualizar el repositorio en la rama principal

Confirme los cambios en el repositorio **Reto1-G<<XX>>** con los comandos **`Commit`** y **`Push`** en la rama **main** local y de GitHub con el comentario *"Avance ordenamientos – Reto 1"* antes de la fecha límite de entrega.

Confirme los cambios en el repositorio **LabSorts-S<<XX>>-G<<YY>>** con los comandos **`Commit`** y **`Push`** en la rama **main** local y de GitHub con el comentario *"Entrega Final – laboratorio 4"* antes de la fecha límite de entrega.

PASO 7: Revisar entregables de la practica

Finalmente, para realizar la entrega del laboratorio revise que sus entregables de la practica estén completos. Para ello, siga las siguientes indicaciones:

1) Acceso al profesor de laboratorio y los monitores de su sección a la organización del grupo
2) Enlace al repositorio GitHub **LabSorts-S<<XX>>-G<<YY>>** con rama ***Main*** actualizada con el comentario *"Laboratorio 4 – Entrega final"* antes del límite de entrega.
1) **README** del repositorio con los datos completos de los integrantes del grupo (nombre completo, correo Uniandes y código de estudiante).
2) Incluir en repositorio la carpeta **`Docs`** el documento **`Observaciones-lab4.docx`** con la siguiente información en formato PDF.

a) Nombres, apellidos, código de estudiante de los integrantes del grupo.
b) Incluir los datos reportados en la Tabla 1, Tabla 2, Tabla 3 y Tabla 4.
c) Respuesta a las preguntas de análisis:

- ¿El comportamiento con relación al orden de crecimiento temporal de los algoritmos es acorde a lo enunciado teóricamente?
- ¿Existe alguna diferencia entre los resultados obtenidos al ejecutar las pruebas en diferentes máquinas?
- De existir diferencias, ¿a qué creen que se deben?
- ¿Cuál Estructura de Datos (ARRAY_LIST o SINGLE_LINKED) funciona generalmente mejor si solo se tiene en cuenta los tiempos de ejecución de los algoritmos?
- Teniendo en cuenta las pruebas de tiempo de ejecución reportadas por los algoritmos de ordenamiento probados (iterativos y recursivos), proponga un listado de estos ordenarlos de menor a mayor teniendo en cuenta el tiempo de ejecución que toma ordenar las obras de arte.

3) Enlace al repositorio GitHub **Reto1-G<<XX>>** con rama **main** actualizada con el comentario *"Avance Pruebas ordenamientos iterativos y recursivos – Reto 1"* antes de la fecha límite de entrega.

PASO 8: Compartir resultados con los evaluadores

Envíe los **enlaces (URL)** de los dos repositorios por **BrightSpace** antes de la fecha límite de entrega.

Recuerden que cualquier documento solicitado durante en la práctica debe incluirse dentro del repositorio GIT y que solo se calificaran los entregables hasta el último **COMMIT** realizado antes de la media noche del miércoles 15 de septiembre.

Anexo: Ejemplo de pruebas exhaustivas

A continuación, se especifica un ejemplo de pruebas exhaustivas de los tiempos de ejecución para los algoritmos de ordenamiento (recursivos e iterativos). Estas pruebas ***NO deben completarse para este laboratorio***, pero SI hacen parte de la entrega del Reto No. 1.

En este caso, se toman los tiempos de ejecución de los algoritmos de ordenamiento estudiados en la clase sobre una lista (ARRAY_LIST y LINKED_LIST) de videos tendencia en YouTube. En las pruebas se varía el tamaño de la lista y se ordenan los elementos basados en el número de reproducciones (views) de cada uno de los videos.

	Máquina de Pruebas
Procesadores	2,3 GHz Intel Core i7 de cuatro núcleos
Memoria RAM (GB)	32 GB
Sistema Operativo	Mac Os Catalina Version 10.15.7

Tabla 5. Especificaciones de la máquina para ejecutar las pruebas de rendimiento.

RESULTADOS PASO 3: EJECUTAR PRUEBAS DE RENDIMIENTO

Tras ejecutar las pruebas como es enunciado en el paso 3 se obtuvieron los datos que se encuentran registrados en las Tablas 6 y 7.

Tamaño de la muestra (ARRAYLIST)	Insertion Sort [ms]	Selection Sort [ms]	Shell Sort [ms]	Quick Sort [ms]	Merge Sort [ms]
1000	458,21	530,70	39,02	33,33	33,51
2000	1794,89	2068,14	73,64	59,66	53,68
4000	7214,92	8106,61	152,44	102,16	105,73
8000	31360,21	33693,65	340,34	205,24	207,79
16000	123395,26	146712,59	1696,85	419,81	418,22
32000	512074,77	614969,52	4216,33	856,29	871,06
64000	1710441,67		10310,33	1863,06	1806,59
128000			25492,56	4273,62	3908,09
256000				9077,16	8566,34
512000					

Tabla 6. Comparación de tiempos de ejecución Vs. Número de elementos para procesar con los algoritmos de ordenamiento utilizando la representación de tipo arreglo.

Tamaño de la muestra (LINKED_LIST)	Insertion Sort [ms]	Selection Sort [ms]	Shell Sort [ms]	Quick Sort [ms]	Merge Sort [ms]
1000	35207,27	30416,33	1785,23	1515,45	178,82
2000	287040,20	252845,32	7385,52	6426,91	638,89
4000			38550,48	26206,65	2576,07
8000			197798,90	499841,78	10671,99
16000					43790,09
32000					172426,58
64000					
128000					
256000					
512000					

Tabla 7. Comparación de tiempos de ejecución Vs. Número de elementos para procesar con los algoritmos de ordenamiento utilizando la representación de lista enlazada.

Es importante resaltar que en la Tabla 6 y la Tabla 7 **NO todos los valores están registrados**, esto significa que las pruebas no se completaron por errores de ejecución al quedarse sin memoria

procesando los datos o porque la prueba llevaba **más de 20 minutos** en ejecución sin devolver un resultado.

Esto depende de las capacidades de la máquina de pruebas, de los tipos de estructuras utilizadas, el número de elementos a ordenar y los datos particulares que se están procesando.

Tomando como base los datos registrados en las Tabla 6 y Tabla 7 se pueden generar diferentes gráficas que nos permiten comprender el funcionamiento de los distintos algoritmos de ordenamiento de acuerdo con la cantidad de datos utilizados y las diferentes implementaciones de la lista.

Por ejemplo, La Ilustración 1 y la Ilustración 2 muestran un resumen de comportamientos en tiempo de todos los algoritmos de ordenamiento según el número de elementos a ordenar y la estructura de datos utilizada (ARRAYLIST y LINKED_LIST).

Por su parte, La Ilustración 3, Ilustración 4, Ilustración 5, Ilustración 6 e Ilustración 7 muestran los tiempos de ejecución de cada uno de los algoritmos de ordenamiento comparativamente a la estructura de datos utilizada. En donde se puede observar que los tiempos varían principalmente por el número de elementos a ordenar y la naturaleza de los datos.

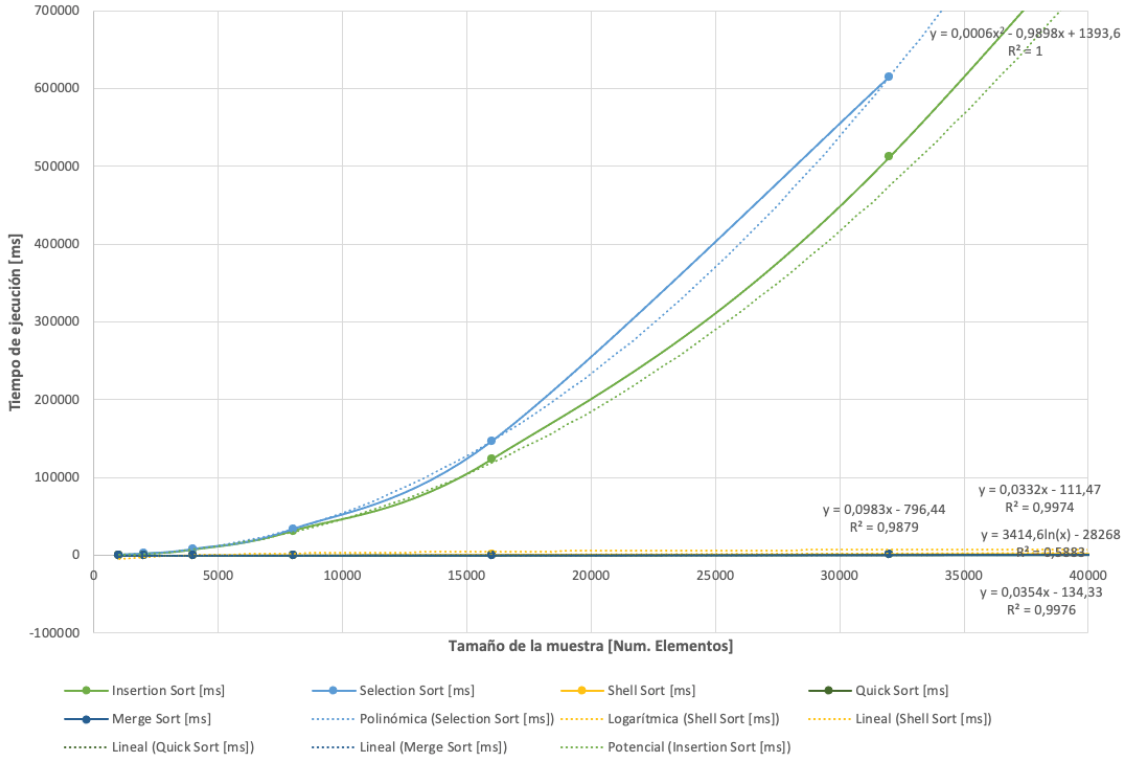

Ilustración 1. Comparación de los tiempos de ejecución para los distintos algoritmos de ordenamiento utilizando la representación de Arreglo.

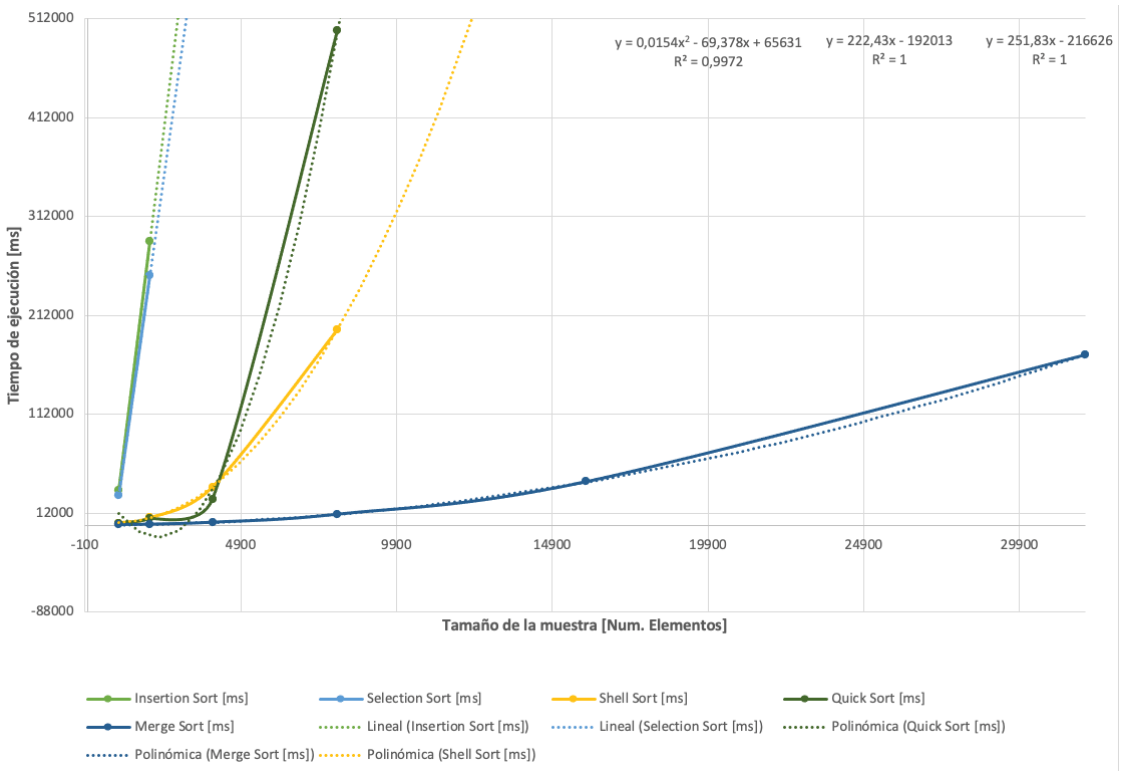

Ilustración 2. Comparación de los tiempos de ejecución para los distintos algoritmos de ordenamiento utilizando la representación de Lista Enlazada.

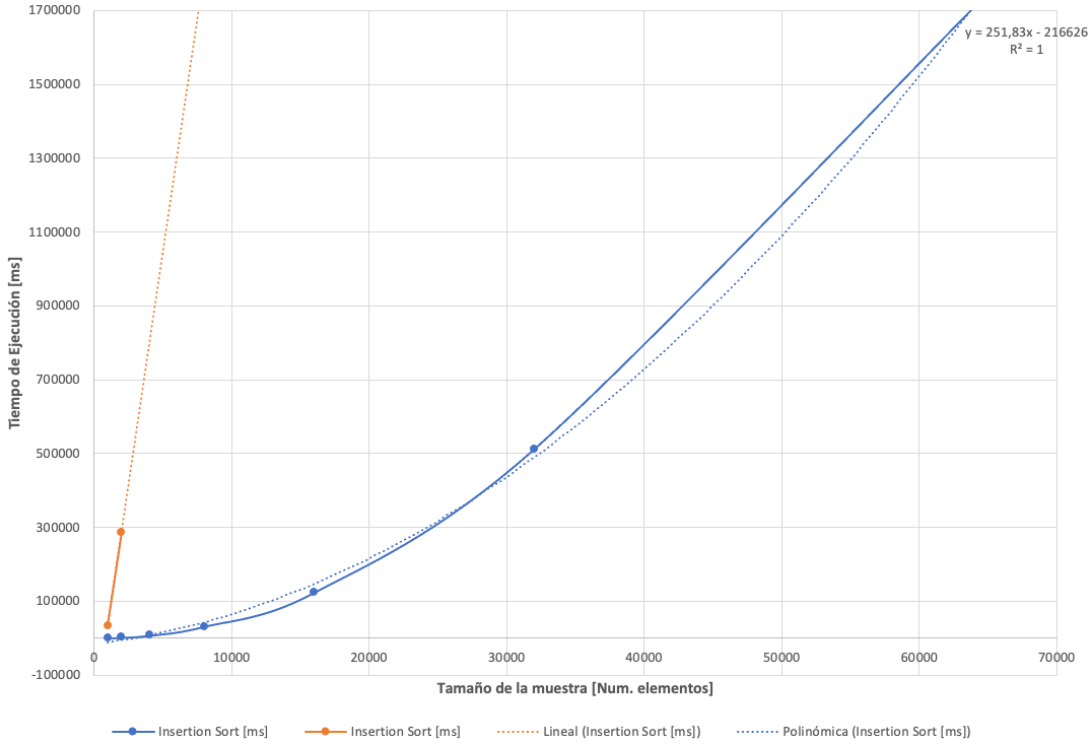

Ilustración 3. Comparación de tiempos de ejecución para el algoritmo Insertion Sort ordenando una lista enlazada (naranja) Vs. Un arreglo (azul).

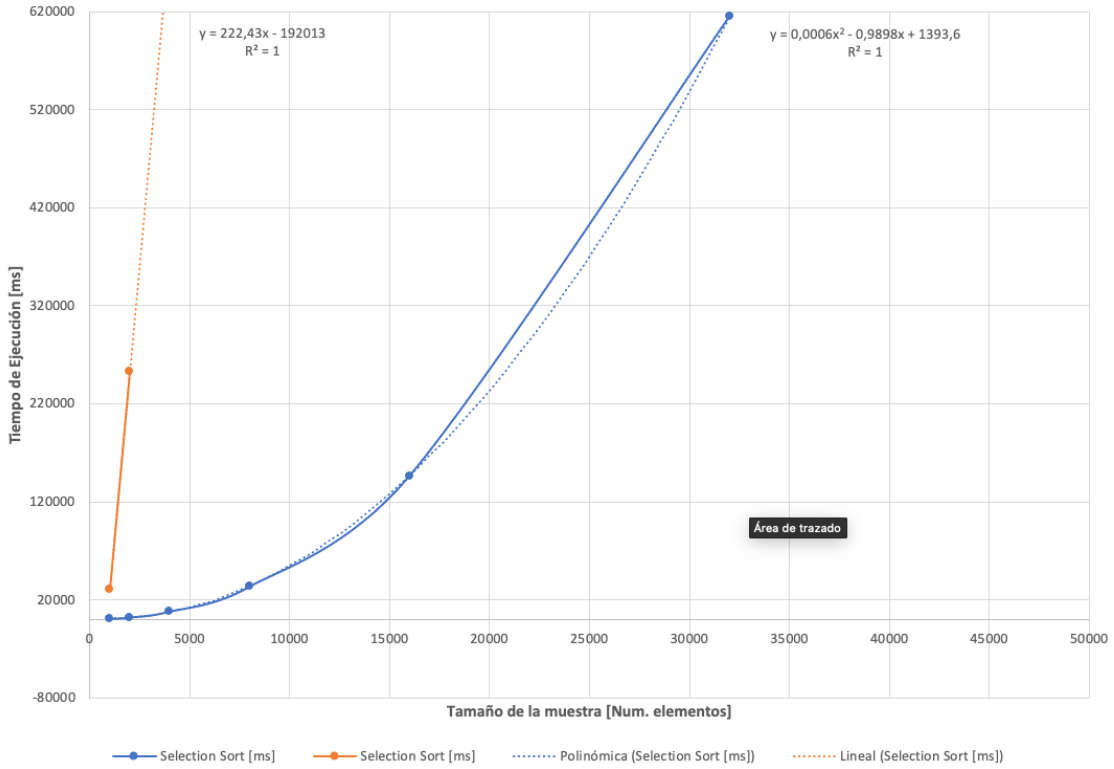

Ilustración 4. Comparación de tiempos de ejecución para el algoritmo Selection Sort ordenando una lista enlazada (naranja) Vs. Un arreglo (azul).

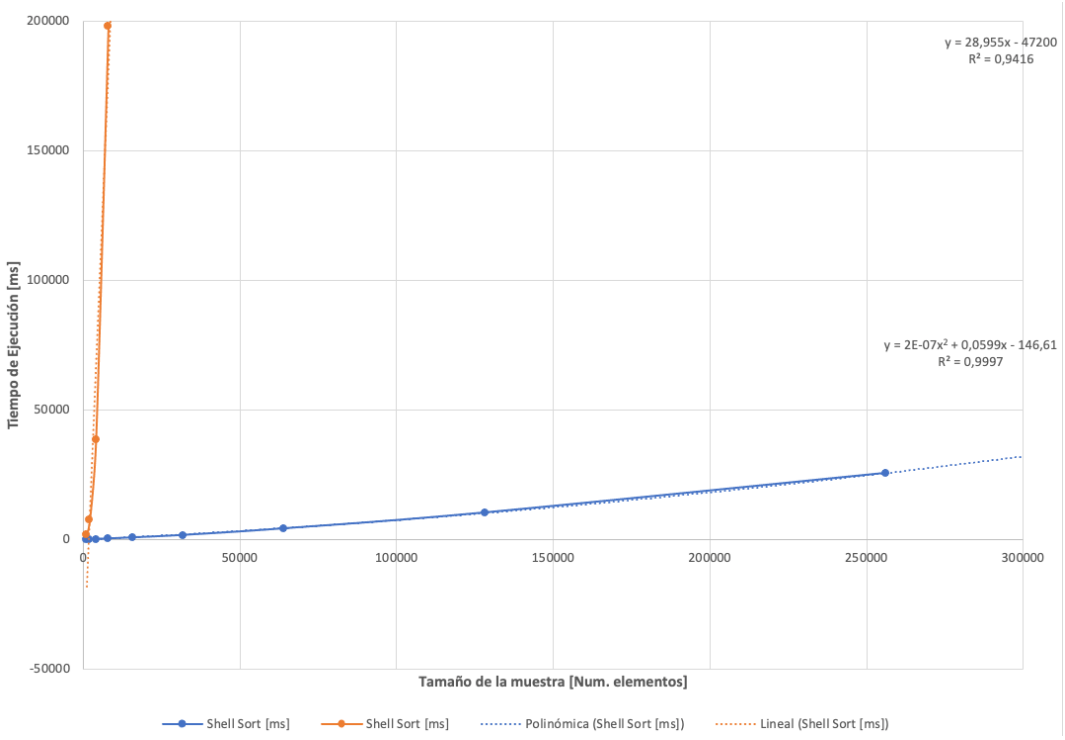

Ilustración 5. Comparación de tiempos de ejecución para el algoritmo Shell Sort ordenando una lista enlazada (naranja) Vs. Un arreglo (azul).

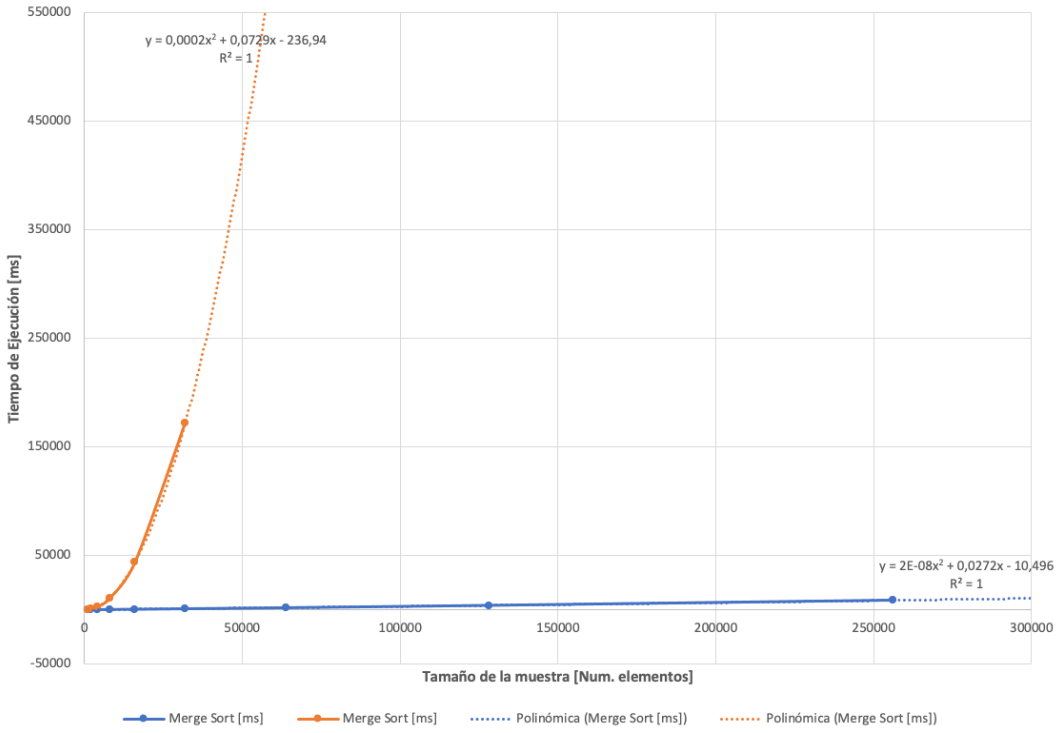

Ilustración 6. Comparación de tiempos de ejecución para el algoritmo Merge Sort ordenando una lista enlazada (naranja) Vs. Un arreglo (azul).

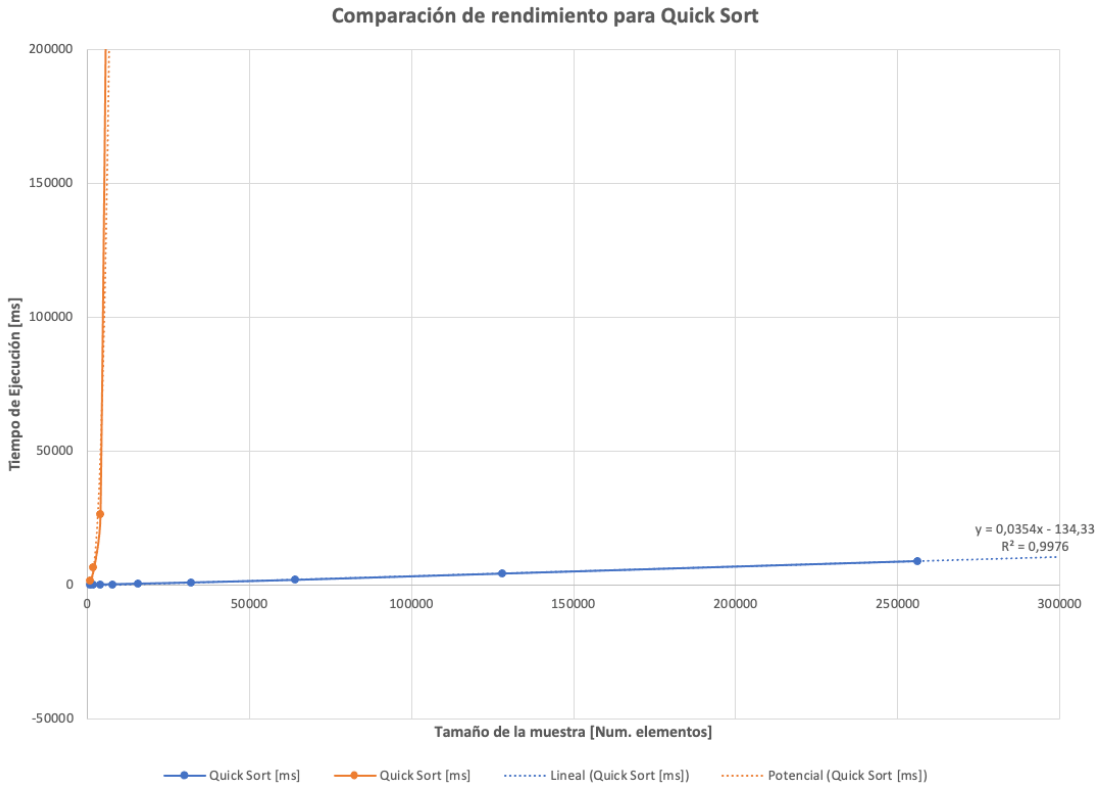

Ilustración 7. Comparación de tiempos de ejecución para el algoritmo Quick Sort ordenando una lista enlazada (naranja) Vs. Un arreglo (azul).