LABORATORIO NO. 5: ALGORITMOS DE ORDENAMIENTO

1 Objetivos

Comprender el funcionamiento y la implementación de los algoritmos de ordenamiento iterativos y recursivo. Al finalizar este laboratorio el estudiante estará en capacidad de:

1) Comprender la complejidad temporal de los algoritmos de ordenamiento.
2) Validar los tiempos de ejecución de los ordenamientos desarrollando pruebas funcionales.
3) Comprender la diferencia entre ordenamientos iterativos y recursivos.
4) Medir y analizar el desempeño de los algoritmos de ordenamiento tomando tiempos de ejecución para evaluar su eficiencia.
5) Comparar ambas implementaciones en términos de tiempo y uso de memoria, identificando en qué casos cada estructura de datos es más eficiente.
6) Determinar en qué escenarios conviene utilizar ArrayList o LinkedList para el ordenamiento de datos, considerando el impacto en rendimiento según el tamaño y naturaleza de los datos.

2 Trabajo Propuesto

2.0 Preparación del ambiente de trabajo

Antes de comenzar por favor siga las siguientes indicaciones:

- El trabajo del laboratorio se basa en el repositorio Laboratorio-5. Para poder comenzar, bifurque el repositorio en su organización siguiendo las convenciones de nombramiento **Laboratorio5-G<<YY>>**, donde **YY** es el número del grupo de trabajo (ej: *Laboratorio5-G01*).

- Clone el repositorio creado en el paso anterior en su computador.

- Un integrante del equipo debe agregar los nombres y usuarios de los integrantes del equipo en el archivo **README** del repositorio. Suba los cambios a GitHub realizando **push** y recuerde que cada estudiante del grupo debe actualizar su repositorio local ejecutando **pull**.

- Un integrante del grupo debe copiar los archivos de estructuras de datos desarrollados en los laboratorios anteriores en las carpetas correspondientes del nuevo repositorio. Recuerde verificar que estos archivos no los copie dentro de la carpeta de Test.

    - **array_list.py** debe ir en **/DataStructures/List/**.
    - **single_linked_list.py** debe ir en **/DataStructures/List/**.
    - **queue.py** debe ir en **/DataStructures/Queue/**.
    - **stack.py** debe ir en **/DataStructures/Stack/**.

    Suba los cambios a GitHub realizando **push** y recuerde que cada estudiante del grupo debe actualizar su repositorio local con **pull**.

- Se recomienda que cada integrante cree una **rama** para trabajar en equipo de manera organizada y simultánea.
- Cada integrante debe descargar los datos de prueba **GoodReads** usados en los laboratorios anteriores. Estos archivos se encuentran en el aula unificada de **BN**. Una vez descargados guárdelos en la carpeta **/Data/GoodReads/** del proyecto.

Recuerde que durante la sesión de laboratorio los miembros del grupo deben completar hasta el paso 2.2.1 y se debe realizar el reléase dentro del horario del laboratorio.

2.1 Implementación de estructuras

En este laboratorio implementará los algoritmos de ordenamiento iterativo (Shell Sort, Insertion Sort y Selection Sort) y recursivo (MergeSort y QuickSort) vistos en el curso, usando la documentación del curso encontrada en **DISC - Data Structures**, como guía para el desarrollo de las estructuras anteriormente mencionadas. Al abrir el enlace de la documentación lo reenviará a una página como la que se presenta en la siguiente captura de pantalla:

Ilustración 1. Documentación DISC

En laboratorios anteriores ya han utilizado la guía de documentación para desarrollar sus implementaciones. Por lo tanto, deben estar familiarizados con cómo emplearla para convertir su contenido en código para laboratorios y retos.

Sin embargo, si necesitan recordarlo, pueden revisar nuevamente el apartado Guía de documentación donde encontrarán una breve explicación sobre la estructura de la documentación del curso y ejemplos de cómo trasladar su contenido al código.

En la sección izquierda de la documentación, encontrarán las estructuras de datos junto con sus implementaciones y componentes fundamentales. Para este laboratorio, trabajarán con ordenamientos por lo que deberán ingresar a la documentación correspondiente, como se muestra en la siguiente captura de pantalla:

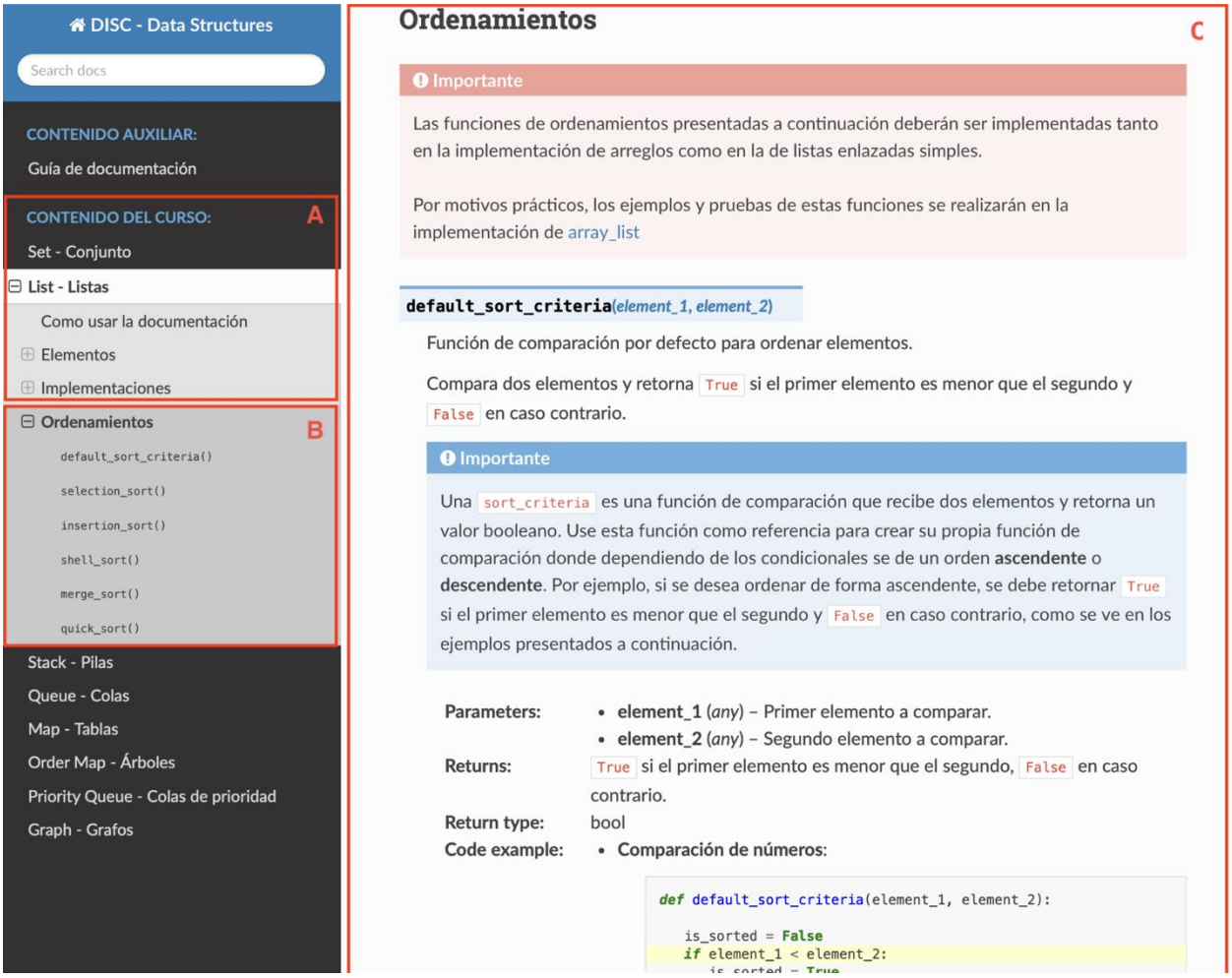

Ilustración 2 Documentación ordenamientos

En la ilustración 2 se pueden identificar tres apartados de la documentación con los cuales se interactuará para identificar los desarrollos de las estructuras.

A. Listado de estructuras y métodos de ordenamiento

Este apartado permite navegar entre las diferentes estructuras de datos vistas en el curso, junto con sus implementaciones y los métodos asociados. En este laboratorio, trabajarán con listas y sus algoritmos de ordenamiento, por lo que deberán revisar las respectivas secciones en la documentación.

B. Archivo de la implementación del algoritmo

Cada algoritmo tiene una función de implementación con un nombre específico que debe ser exactamente el mismo que aparece en la documentación. De lo contrario, los tests pueden no funcionar correctamente. Para este laboratorio, deberán crear las siguientes funciones:

- **default_sort_criteria()**
- **selection_sort()**
- **insertion_sort()**
- **shell_sort()**
- **merge_sort()**
- **quick_sort()**

Recuerde que debe crear dos implementaciones de cada función una por cada TAD de lista vista en el curso. Las funciones debe realizarlas dentro de los archivos array_list.py y single_linked_list.py archivos que se encuentran dentro de la carpeta **DataStructures/List** y que usted importo previamente:

C. Funciones que implementar

En el detalle de cada archivo encontrarán las funciones que deben desarrollar para completar la implementación. La documentación describe cada función, los parámetros que recibe y el valor de retorno esperado. Cada una de estas funciones deberá implementarse respetando la estructura y los parámetros indicados en la documentación. Asegúrense de revisar la descripción para garantizar el correcto funcionamiento de su código.

2.2 Implementación de Algoritmos

Después de clonar el repositorio en su máquina local (paso que debió completar en una sección 2.0 Preparación del ambiente de trabajo de este laboratorio), encontrará una estructura de proyecto similar a la mostrada en la Ilustración 3. La organización del proyecto sigue el mismo esquema utilizado en laboratorios anteiorres, por lo que la carpeta DataStructures debería verse de la siguiente manera:

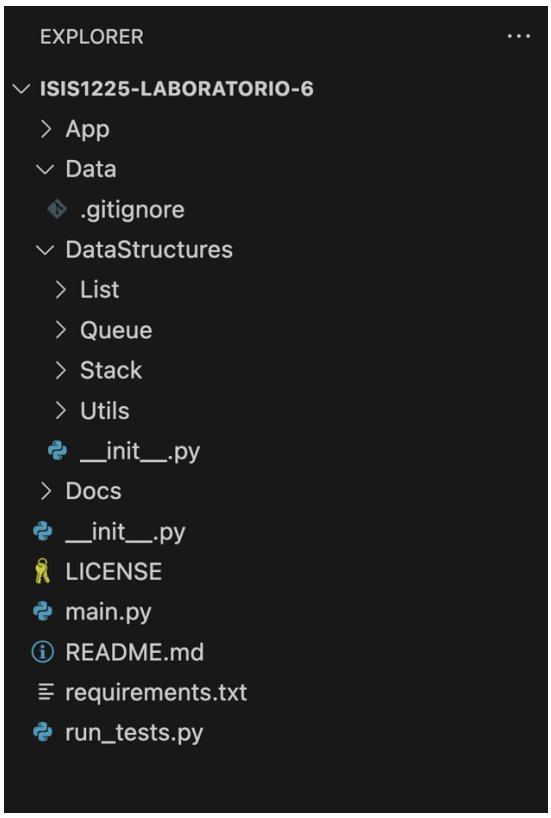

Ilustración 3 estructura proyecto

Para este laboratorio, deberá modificar las implementaciones de *array_list.py* y *single_linked_list.py*, incorporando las funciones de ordenamiento. Cada tipo de lista debe contar con su propia implementación de estas funciones.

A partir de este punto, el objetivo es que, como equipo, sigan desarrollando las estructuras de datos vistas en el curso, consolidando su implementación para su aplicación en los próximos retos.

2.2.1 Implementación algoritmos de ordenamiento iterativo en array list

Siguiendo la documentación de DISC - Data Structures, en este momento se encargarán de desarrollar los algoritmos de ordenamiento iterativo **selection_sort()**, **insertion_sort()** y **shell_sort()**así como de crear la función **default_sort_criteria().**

Para iniciar el laboratorio, abra el archivo **array_list.py** y comience por desarrollar la función **default_sort_criteria()** como primer paso. Siga la documentación proporcionada, donde encontrará un código de ejemplo que debe implementar. Esta función debe quedar exactamente como se presenta en la documentación. Una vez completada esta función, diríjase al archivo de **single_linked_list.py** y haga la misma función para que funcione con este tipo de listas. Haga push al repositorio y continúe con la implementación de los algoritmos de ordenamiento.

Se recomienda dividir el trabajo entre los integrantes del grupo, partiendo de este punto común donde todos ya cuentan con la función de comparación implementada en ambos archivos. Cada estudiante puede encargarse de un algoritmo específico para el archivo array_list.py, recuerde que al dividirse cada uno los ira desarrollándo en su propia rama:

- Est-1: **selection_sort()**
- Est-2: **insertion_sort()**
- Est-3: **shell_sort()**

Una vez desarrollados los algoritmos, recuerden que se han creado una serie de pruebas unitarias mínimas para verificar el correcto funcionamiento de estas funciones. Estas pruebas están diseñadas para evaluar que cada algoritmo se comporte según lo esperado.

Los tests se encuentran en la ruta **DataStructures/List/Test** como se observa en la estructura mostrada en la ilustración 4.

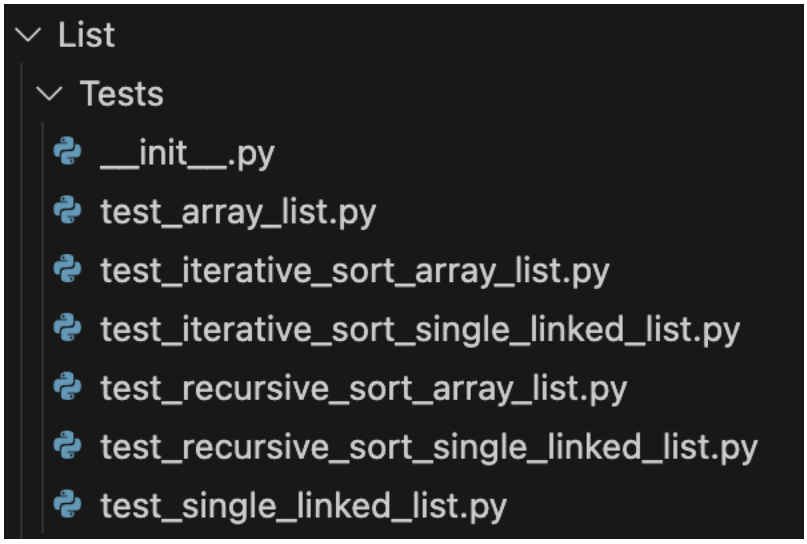

Ilustración 4 estructura test

Abra los archivos **test_iterative_sort_array_list.py** y **test_iterative_sort_single_linked_list.py**, para familiarizarse con los tests. Estos archivos contienen todas las pruebas unitarias diseñadas para los algoritmos de ordenamiento iterativo. De igual manera existen otros dos archivos test para evaluar los algoritmos de ordenamiento recursivo los cuales se desarrollarán más adelante en el laboratorio. Como se puede observar en la primera línea de código, los tests importan el archivo correspondiente según lo definido en la documentación. Es **fundamental** que tanto el nombre del archivo como el de las funciones y el orden de los parámetros coincidan con la documentación oficial para garantizar la correcta ejecución de los tests.

Recuerde que cada vez que se desarrolle una nueva función de ordenamiento y se desee verificar su funcionamiento, puede ejecutar los tests. Esto lo puede lograr ejecutando el siguiente comando según su sistema operativo y la configuración de su entorno, en la terminal desde la carpeta raíz del repositorio:

- En Windows: python .\run_tests.py
- En macOS o Linux: python3 run_tests.py

Tenga en cuenta que dependiendo de cómo haya instalado Python en su sistema, puede necesitar utilizar python, python3 o incluso py. Esta configuración debió haber sido identificada en los primeros laboratorios, así que utilice el comando que funcione correctamente en su entorno.

Al ejecutar el código anterior, encontrará el menú mostrado en la siguiente imagen, seleccionando la opción **5.A**, podrá ejecutar las pruebas de los algoritmos de ordenamiento iterativo.

```
====================== Bienvenido a las pruebas de EDA ======================
1. Todas las estructuras
2. Listas
     2.A Lista de arreglos
     2.B Lista encadenadas
3. Colas (Queues)
4. Pilas (Stacks)
5. Métodos de Ordenamiento
     5.A Ordenamientos Iterativos
     5.B Ordenamientos Recursivos
0. Salir
Ingrese el número de la opción que desea ejecutar:
5.b
```

Ilustración 5 menú tests

Al ejecutar las pruebas, debería aparecer una palabra en verde al lado de cada test indicando PASSED, lo que confirma que su algoritmo está funcionando correctamente. Aquellos algoritmos que aún no han sido implementados aparecerán como SKIPPED, resaltados en amarillo. Si se presenta un error diferente, verifique que los nombres de los archivos sean correctos y que todas las estructuras desarrolladas hasta el momento hayan sido copiadas en las carpetas correspondientes.

Una vez los tres integrantes del grupo hayan verificado que sus respectivos algoritmos pasan satisfactoriamente las pruebas unitarias de forma individual, deben subir los cambios uno por uno a la rama main. Recuerden actualizar la rama antes de realizar cada *merge*. Una vez que todos los cambios estén integrados en main, ejecuten nuevamente las pruebas para confirmar que el proyecto funciona correctamente de forma conjunta.

En este momento con las funciones de **default_sort_criteria()** en ambos archivos y **selection_sort()**, **insertion_sort()** y **shell_sort()** implementadas en ArrayList, al ejecutar las pruebas con la opción 5.A el resultado debería ser igual al mostrado en la Ilustración 6.

```
Ingrese el número de la opción que desea ejecutar:
5.A
============================================= test session starts ==============================================
platform darwin -- Python 3.13.2, pytest-8.3.4, pluggy-1.5.0 -- /Users/Lina/.local/pipx/venvs/pytest/bin/python
cachedir: .pytest_cache
rootdir: /Users/Lina/Library/Mobile Documents/com~apple~CloudDocs/Universidad/EDA/ EDA 2025-01/Lab 6/ISIS1225-Solución-Lab-6
collected 56 items / 52 deselected / 4 selected

DataStructures/List/Tests/test_iterative_sort_array_list.py::test_default_sort_criteria PASSED                  [ 25%]
DataStructures/List/Tests/test_iterative_sort_array_list.py::test_selection_sort PASSED                         [ 50%]
DataStructures/List/Tests/test_iterative_sort_array_list.py::test_insertion_sort PASSED                         [ 75%]
DataStructures/List/Tests/test_iterative_sort_array_list.py::test_shell_sort PASSED                             [100%]

========================================== 4 passed, 52 deselected in 0.03s ==========================================
sh: py: command not found
sh: python: command not found
/opt/homebrew/opt/python@3.13/bin/python3.13: No module named pytest
============================================= test session starts ==============================================
platform darwin -- Python 3.13.2, pytest-8.3.4, pluggy-1.5.0 -- /Users/Lina/.local/pipx/venvs/pytest/bin/python
cachedir: .pytest_cache
rootdir: /Users/Lina/Library/Mobile Documents/com~apple~CloudDocs/Universidad/EDA/ EDA 2025-01/Lab 6/ISIS1225-Solución-Lab-6
collected 56 items / 52 deselected / 4 selected

DataStructures/List/Tests/test_iterative_sort_single_linked_list.py::test_default_sort_criteria PASSED          [ 25%]
DataStructures/List/Tests/test_iterative_sort_single_linked_list.py::test_selection_sort SKIPPED (selection_sort()...) [ 50%]
DataStructures/List/Tests/test_iterative_sort_single_linked_list.py::test_insertion_sort SKIPPED (insertion_sort()...) [ 75%]
DataStructures/List/Tests/test_iterative_sort_single_linked_list.py::test_shell_sort SKIPPED (shell_sort() is not ...) [100%]

==================================== 1 passed, 3 skipped, 52 deselected in 0.02s =====================================
sh: py: command not found
sh: python: command not found
/opt/homebrew/opt/python@3.13/bin/python3.13: No module named pytest
====================== Gracias por ejecutar las pruebas ======================
```

Ilustración 6 ejecución test iterativos array list

Finalmente, guarden y suban los cambios de la clase con el commit “Entrega inicial Laboratorio 6” a la rama main del repositorio. Cree un release con el título “Entrega Inicial Laboratorio 6” y el tag “0.0.1” en su repositorio.

2.2.2 Implementación de los algoritmos de ordenamiento restantes

A partir de este punto, queda a criterio del equipo cómo organizar y dividir el trabajo. Se recomienda continuar utilizando ramas individuales para cada integrante, lo cual permite evitar conflictos al momento de integrar los cambios y mantener un control claro sobre el desarrollo. Recuerden siempre actualizar y sincronizar con la rama **main** antes de hacer un merge, y validar que las pruebas sigan pasando correctamente tras cada integración.

Continúen ahora con la implementación de los algoritmos de ordenamiento iterativo en el archivo **single_linked_list.py**. Luego, realicen las implementaciones de los algoritmos de ordenamiento recursivo **merge_sort** y **quick_sort** tanto para **ArrayList** como para **SingleLinkedList**. Al finalizar todas las implementaciones, y haber hecho merge en main ejecuten las pruebas utilizando la opción 5 del menú. Esta opción ejecutará todos los tests relacionados con los algoritmos de ordenamiento. Al ejecutar las pruebas unitarias, asegúrese de que todas se muestren completamente en color verde con la indicación **PASSED**. Esto confirmará que los algoritmos de ordenamiento en **ArrayList** y **SingleLinkedList** funcionan correctamente y han superado todas las pruebas sin errores.

No obstante, recuerde que estas pruebas son mínimas y no garantizan por sí solas que su implementación sea completamente correcta. Es responsabilidad suya verificar manualmente que las estructuras estén bien implementadas y que el comportamiento de los algoritmos sea el esperado.

NOTA: En algunos casos experimentales Python y el IDE pueden declarar que se alcanzó el límite de recursión con un mensaje del estilo de **“RecursionError: maximum recursion depth exceeded in comparison”.** En este caso se recomienda actualizar el límite de recursión del programa desde el **view.py** con el código escrito en Segmento 1

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

Segmento 1. modificaciones para el límite de recursión de Python en el view.py.

Confirme los cambios del avance de laboratorio siguiendo los pasos aprendidos en prácticas anteriores (**Commit** y **Push**) y utilice el comentario “Algoritmos de ordenamiento completos laboratorio 6”.

2.3 Implementación de funcionalidad

Ahora pasamos a la funcionalidad de la aplicación para ello debe completar las funciones que se encuentran incompletas en el archivo logic.py y en view.py como se presenta en la imagen de referencia, para que la aplicación quede completamente implementada.

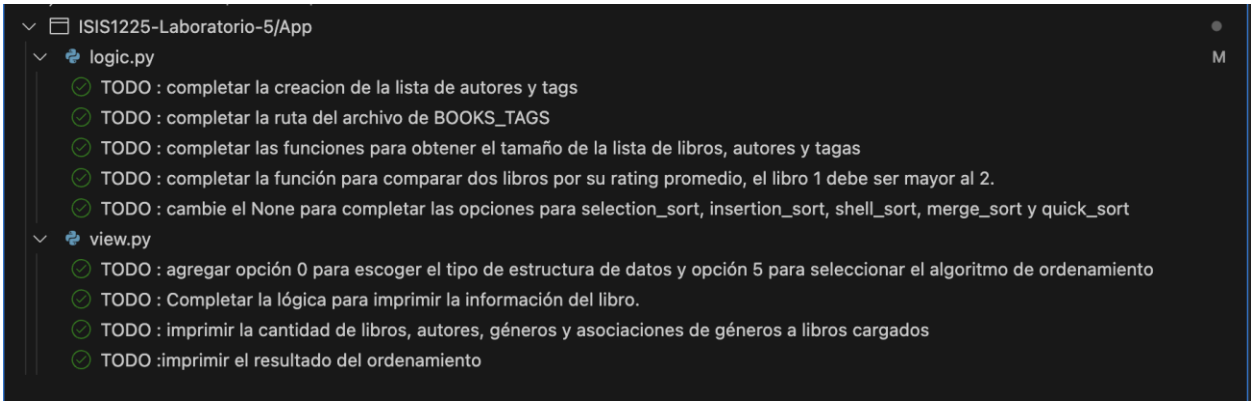

Ilustración 9 TODO

2.4 Preparar las pruebas de tiempos de ejecución

Deberán entregar un informe y archivos de Excel con los datos de rendimiento obtenidos. La plantilla para estos informes se encuentra en la carpeta "Docs" del proyecto. Antes de iniciar las pruebas de esta práctica recomendamos tener en cuenta las siguientes instrucciones:

- Diligencien la Tabla 1 con la información del computador designado donde se ejecutará las pruebas de los algoritmos de ordenamiento ubicado en el documento **Observaciones-lab5.docx**.

	Máquina 1
Procesadores	
Memoria RAM (GB)	
Sistema Operativo	

Tabla 1. especificaciones de la máquina para ejecutar las pruebas de rendimiento.

Por ejemplo:

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

Tabla 2. ejemplo especificaciones de la máquina para ejecutar las pruebas de rendimiento.

- Ejecute todas las pruebas de rendimiento propuestas en el documento con el archivo de datos “large”.
- 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**).
- Ejecute cada prueba por lo menos **tres (3) veces** para obtener un promedio del tiempo de ejecución consistente. Evitando errores e interrupciones dentro de la misma máquina, para luego promediar el resultado y registrarlo en las tablas correspondientes.
- Registre cada uno de estos tiempos en **milisegundos (ms)** y con 2 cifras decimales redondeado hacia arriba desde la mitad. Ej: 43494.4985 ms se aproxima a 43494.50 ms.

NOTA 1: Recuerde que si la ejecución de un algoritmo de ordenamiento tarda 15 minutos o más, debe dar por terminada la prueba y continuar con la verificación de otro algoritmo. En ese caso, se recomienda revisar cuidadosamente la implementación, ya que lo más probable es que el algoritmo no esté funcionando correctamente. Intente identificar posibles cuellos de botella o errores lógicos que estén generando el retraso en la ejecución. Revise especialmente condiciones de parada, bucles infinitos, o comparaciones mal definidas que puedan estar afectando el rendimiento.

En esta fase del laboratorio, se evaluará el rendimiento de los todos algoritmos de ordenamiento en **ArrayList** y **Single Linked List**, midiendo sus tiempos de ejecución para analizar su eficiencia.

Por ahora, **ignore la sección de comparación de tiempos de los mejores algoritmos** que se encuentra en su archivo de observación. Solo podrá completar esta información cuando se le indique.

2.5 Procedimiento tiempo de ejecución ordenamientos

Dirijase a su aplicación y ejecútela desde el main, podrá observar un menú como el de la ilustración 10.

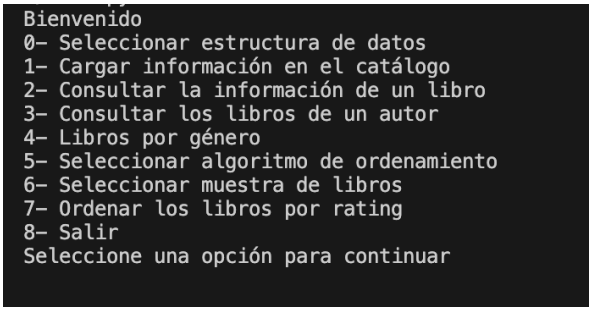

Ilustración 5 Menú de opciones

Para realizar las mediciones de tiempos ejecute el menú como se le especifica a continuación:

Antes que nada recuerde correr el programa utilizando el archivo "large".

1. **Selección de la estructura de datos (Opción 0)**
    - Se permite al usuario elegir entre **ArrayList** y **Single Linked List**, cada una con características distintas en términos de acceso y manipulación de datos.
2. **Carga de información (Opción 1)**
    - Se almacenan los libros en la estructura de datos seleccionada, garantizando que puedan ser procesados correctamente.
3. **Selección del algoritmo de ordenamiento (Opción 5)**
    - Se le permite escoger cualquier algoritmo de ordenamiento visto.
4. **Selección del tamaño de la muestra (Opción 6)**
    - Se define el número de elementos sobre los cuales se ejecutará la prueba de ordenamiento. **Recuerde ingresar directamente la cantidad de datos, no el porcentaje.** Por ejemplo, si desea trabajar con el **10 % de los datos** y el total de registros es **1000**, debe ingresar el valor **100**, que corresponde al 10 % de 1000.
5. **Ejecución del ordenamiento y medición del tiempo (Opción 7)**

- Ejecuta el ordenamiento seleccionado con la cantidad de datos que especifico recuerde registrar el tiempo de ejecución del algoritmo en la estructura seleccionada para su posterior análisis.

Repita este proceso para cada una de las cantidades de datos definidas, así como para cada uno de los algoritmos de ordenamiento. De esta manera, podrá completar correctamente todas las tablas correspondientes a las pruebas de desempeño de los distintos algoritmos.

2.6 Comparar Algoritmos y Estructuras

Después de registrar los datos de los algoritmos de ordenamiento, como grupo deberá graficar los resultados para comparar el comportamiento de los algoritmos y responder preguntas de análisis de la parte 1. Para ello, deberán utilizar el archivo "ISIS1225 – Tablas de Datos Lab 5.xlsx" para registrar la información.

Este archivo contiene un total de 9 pestañas. Es importante que siga el paso a paso propuesto durante el laboratorio para completar el análisis correctamente. No anticipe información en pestañas que aún no han sido solicitadas. Seguir la secuencia indicada le permitirá avanzar de forma clara y obtener mejores resultados en el análisis.

1. Complete únicamente las tablas tituladas "TIEMPOS DE EJECUCIÓN ARRAY LIST [ms]" y "TIEMPOS DE EJECUCIÓN SINGLE LINKED LIST [ms]", ubicadas en la pestaña *01-Data*. Por ahora, ignore las demás pestañas, ya que las gráficas comparativas de cada algoritmo de ordenamiento se generarán automáticamente a partir de la información que usted registre en estas tablas.

2. Una vez diligenciadas las tablas en la pestaña **01-Data**, diríjase a cada una de las demás pestañas, **excepto la pestaña 09-Bests**. Notará que se han generado automáticamente las gráficas comparativas con base en los datos ingresados.

   A continuación, una breve explicación de lo que encontrará en cada una:

   - **01-Data**: Contiene las tablas base de datos de ejecución. Toda la información registrada aquí alimenta las gráficas del resto de pestañas.
   - **02-Array List** y **03-Single Linked List**: Comparan todos los algoritmos entre sí dentro de cada tipo de lista (estructura ArrayList y SingleLinkedList, respectivamente).
   - **04-Insertion Sort**, **05-Selection Sort**, **06-Shell Sort**, **07-Merge Sort**, **08-Quick Sort**: Comparan la ejecución del mismo algoritmo entre ArrayList y SingleLinkedList, es decir cómo se comporta cada algoritmo según la estructura de datos utilizada.

3. Compare los resultados obtenidos en las máquinas de trabajo del grupo con la complejidad teórica de cada algoritmo. Para ello, genere las líneas de tendencia en las gráficas de cada pestaña, desde la 04 hasta la 08. Esto le permitirá visualizar el comportamiento empírico de cada algoritmo y contrastarlo con su complejidad esperada (por ejemplo, O(n), O(n log n), $O(n^2)$, etc.).

4. Con esta información, reúnanse como grupo para discutir y responder las preguntas de análisis parte 1 del documento en Word.

2.7 Comparar tiempos de ejecución de los mejores algoritmos

Discuta con sus compañeros y analicen cuál es el algoritmo que presenta mejor eficiencia. En las preguntas de análisis parte 1 debió identificar el mejor algoritmo de ordenamiento iterativo y el mejor recursivo según su criterio.

Ahora, complete **cada una de las tablas** en la sección **"Comparación de tiempos de los mejores algoritmos"** del documento de observaciones. Recuerde que ya debió registrar estos datos, así que puede copiar y pegar los valores obtenidos en las tablas anteriores.

Después de registrar los datos en el documentó de observaciones, diríjase al archivo Excel y registre los datos en la sección TIEMPOS DE EJECUCIÓN MEJORES ALGORITMOS [ms] de la hoja 01- Data. Finalmente, analice la gráfica obtenida en la pestaña 09 para evaluar los resultados, y genere las líneas de tendencia adecuadas para cada algoritmo.

En este momento, ya cuenta con los datos necesarios y el análisis correspondiente para responder las preguntas que se encuentran al final del documento (preguntas de análisis parte 2).

2.8 Documentar el análisis de los ordenamientos

Al completar las pruebas, y responder las preguntas, verifique que el docmeunto **Observaciones-lab 5.docx** incluya lo siguiente:

a. Nombres, apellidos, código de estudiante de los integrantes del grupo.
b. Todas las tablas deben estar completas.
c. El grupo incluye todas las respuestas a las preguntas de análisis.

OJO: Aunque se proporciona el formato del documento de observaciones en Word, es necesario que conviertan y entreguen el archivo final en PDF.

Para garantizar la correcta entrega de los archivos, cada grupo debe guardar y añadir los documentos correspondientes en la carpeta Docs del repositorio desde la rama main. Se espera que, al finalizar el laboratorio, la carpeta contenga el archivo de Excel con los datos reportados, así como un documento en formato PDF titulado **Observaciones-Lab6.pdf**, el cual compila las tablas de rendimiento y las respuestas a las preguntas de análisis.

Recuerde que para realizar esta carga, usted puede simplemente arrastrar los archivos al explorador de archivos de su computador o desde Visual Studio Code, asegurándose de ubicarlos dentro de la carpeta **Docs**. Automáticamente se mostrarán como archivos modificados o nuevos.

3 Entrega

3.1 Confirmar cambios finales en el laboratorio

Asegúrese que main quedo actualizada con los últimos cambios y que estos se vean reflejados en GitHub, donde también deben salirle todos los documentos de la carpeta Docs un Excel y un pdf.

Cree un release con el título “Entrega Final Laboratorio 6” y el tag 1.0.0 antes de la fecha límite de entrega del laboratorio

3.2 Compartir resultados con los evaluadores

Envíe el **enlace (URL)** del repositorio 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 que estén dentro del último **RELEASE** realizado previo a la Fecha/Hora Límite de Entrega indicada al inicio de este enunciado. No se recibirá ninguna entrega fuera de bloque neón.