LABORATORIO NO. 6: ALGORITMOS DE ORDENAMIENTO RECURSIVO

1 Objetivos

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

1) Comprender la complejidad temporal de los algoritmos de ordenamiento recursivos.
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 Merge Sort y Quick Sort en ArrayList y LinkedList, 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

Antes de comenzar por favor siga las siguientes indicaciones:

- El trabajo del laboratorio se basa en el proyecto Laboratorio-6. Para poder comenzar, bifurque el repositorio en su organización siguiendo las convenciones de nombramiento **Laboratorio6-G<<YY>>**, donde **YY** es el número del grupo de trabajo (ej.: *Laboratorio6-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 con **pull**.
- Copie los archivos de estructuras de datos previos 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** en el repositorio 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** y 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 recursivo vistos en el curso (MergeSort y QuickSort), 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 recursivos merge_sort() y quick_sort(), por lo que deberán ingresar a la documentación correspondiente, como se muestra en la siguiente captura de pantalla:

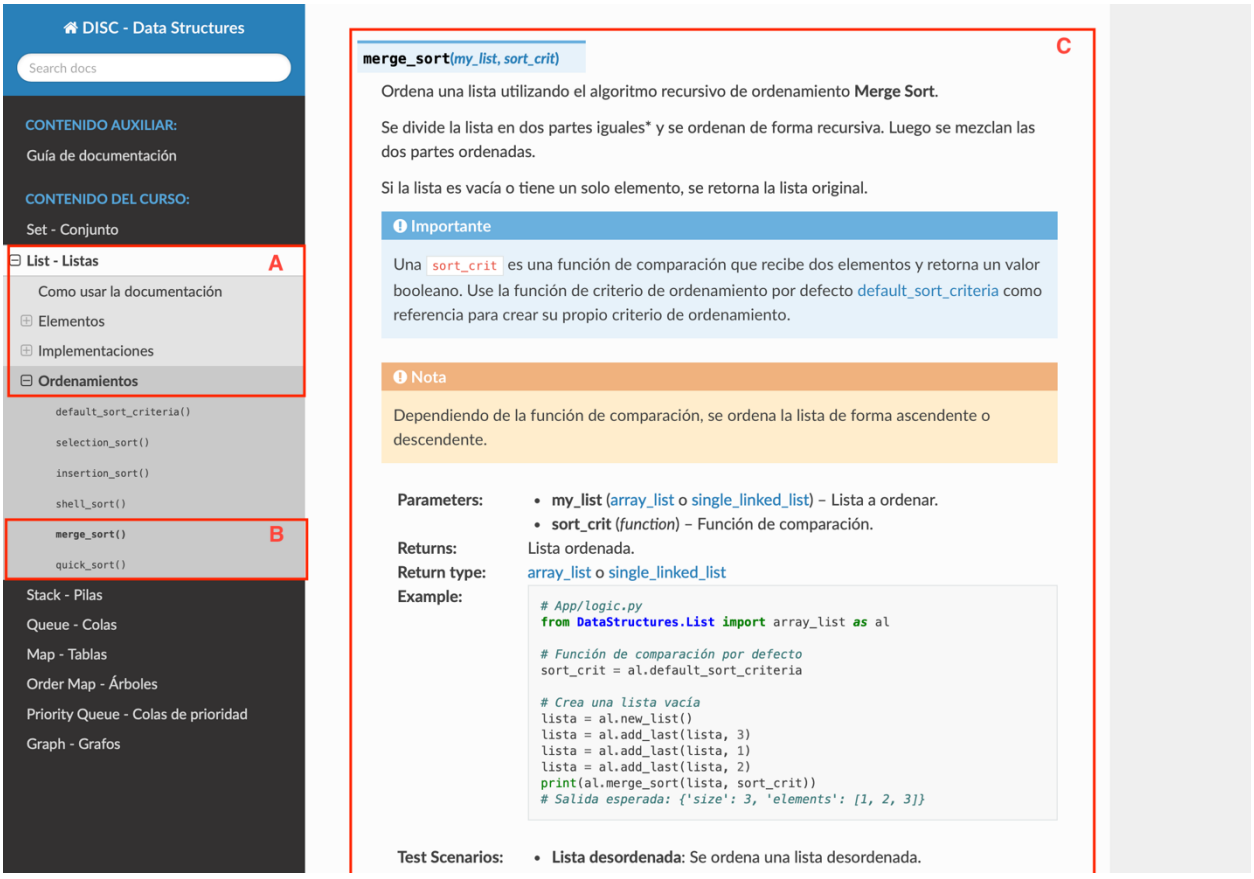

Ilustración 2 Documentación ordenamientos

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:

- **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**:

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 al inicio del laboratorio) encontrará una estructura de proyecto similar a la mostrada en la ilustración3. La organización del proyecto sigue el mismo esquema trabajado en el Laboratorio 5, por lo que la carpeta *DataStructures* debe contener las subcarpetas *List*, Stack y Queue dentro de la cual se encuentran las implementaciones correspondientes desarrolladas en laboratorios anteriores.

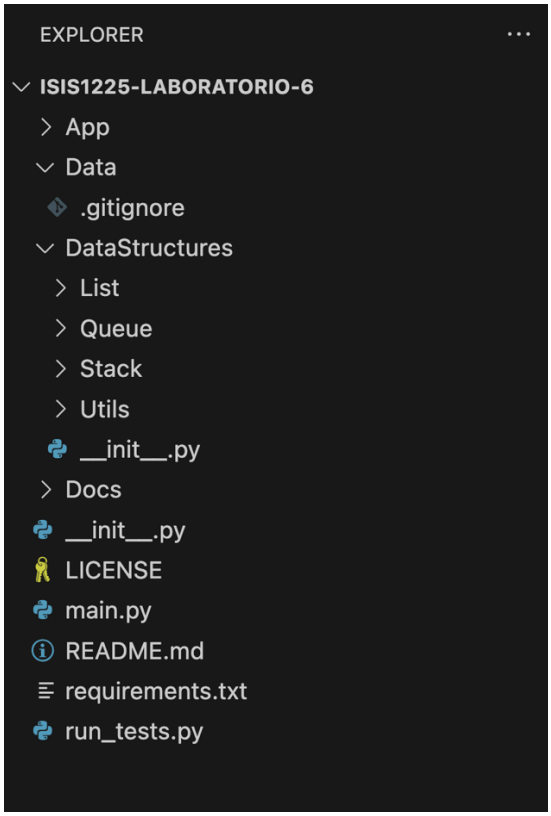

Ilustración 3 estructura proyecto

Para este laboratorio, deberá modificar las implementaciones de *array_list.py* y *single_linked_list.py*, incorporando nuevas 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 recursivo en array_list

Siguiendo la documentación de **DISC - Data Structures**, en este momento se encargarán de desarrollarán los algoritmos de ordenamiento recursivo **Merge Sort** y **Quick Sort** en ArrayList. Para verificar el correcto funcionamiento de estas funciones, se han creado una serie de pruebas unitarias mínimas diseñadas para evaluar su comportamiento esperado.

Es importante destacar que estas pruebas **no evalúan la exactitud de la implementación ni la complejidad temporal del código**, sino únicamente su funcionamiento básico. Esto significa que una función podría no ordenar correctamente o presentar una complejidad ineficiente, pero aun así pasar las pruebas.

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_recursive_sort_array_list.py** y **test_recursive_sort_single_linked_list.py**, para familiarizarse con los tests. Estos archivos contienen todas las pruebas unitarias diseñadas para los algoritmos de ordenamiento recursivo. 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 con el siguiente comando en la terminal desde la carpeta raíz del repositorio:

```
ISIS1225-Laboratorio-6 % python .\run_tests.py
```

Ilustración 5

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

```
====================== 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 6 menú tests

En este momento, con las funciones quick_sort() y merge_sort() implementadas en ArrayList, al ejecutar las pruebas, el resultado debería ser similar al mostrado en la Ilustración 6.

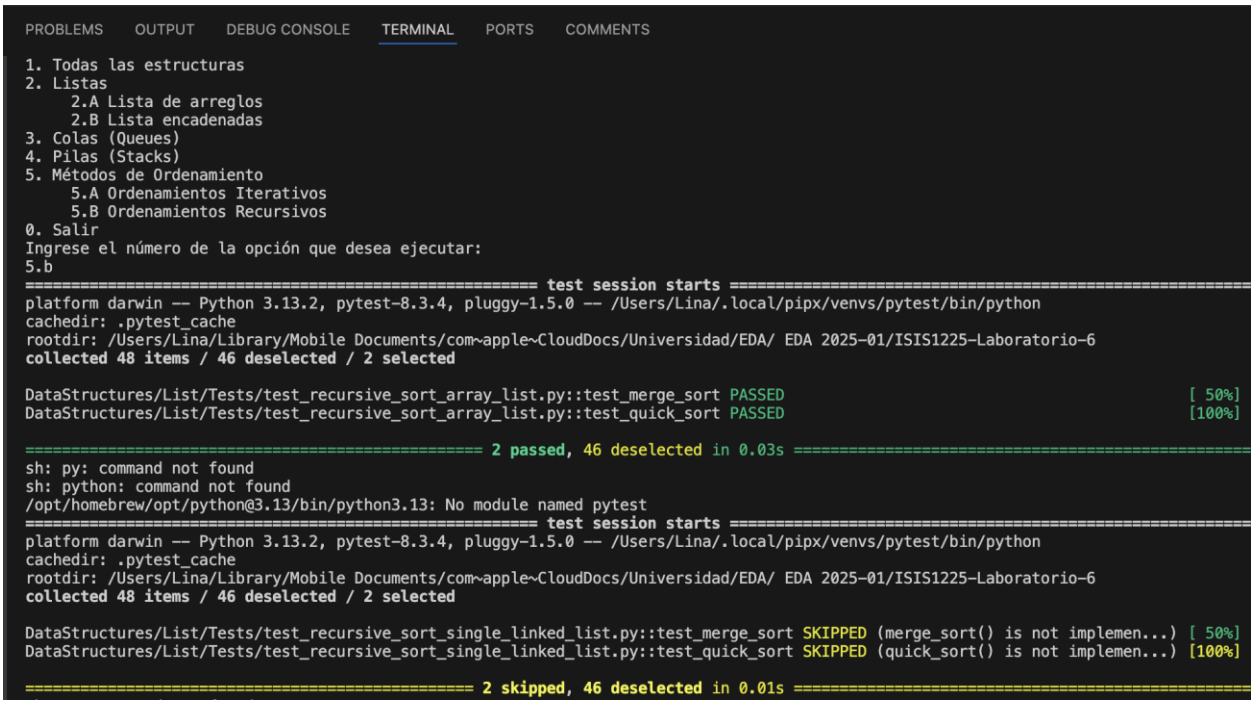

Ilustración 7 ejecución test array list

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 algoritmos de ordenamiento recursivo en single_linked_list

Continue ahora trabajando en la implementación de los algoritmos de ordenamiento recursivo merge sort y quick sort pero en single linked list. Al finalizar la implementación, verifique por su cuenta que las estructuras estén correctamente implementadas. Además, al ejecutar las pruebas unitarias, asegúrese de que todas se muestren completamente en color verde, tal como se presenta en la ilustración 8. Esto indicará que las funciones quick_sort() y merge_sort() en ArrayList funcionan correctamente y han pasado todas las pruebas sin errores.

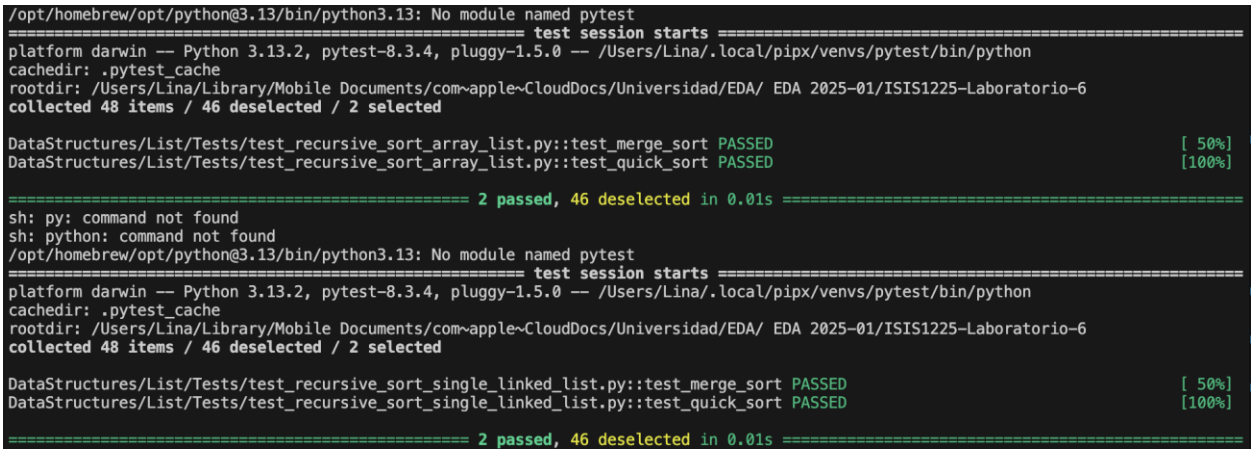

Ilustración 8 ejecución test completos

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 recursión Laboratorio – 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.

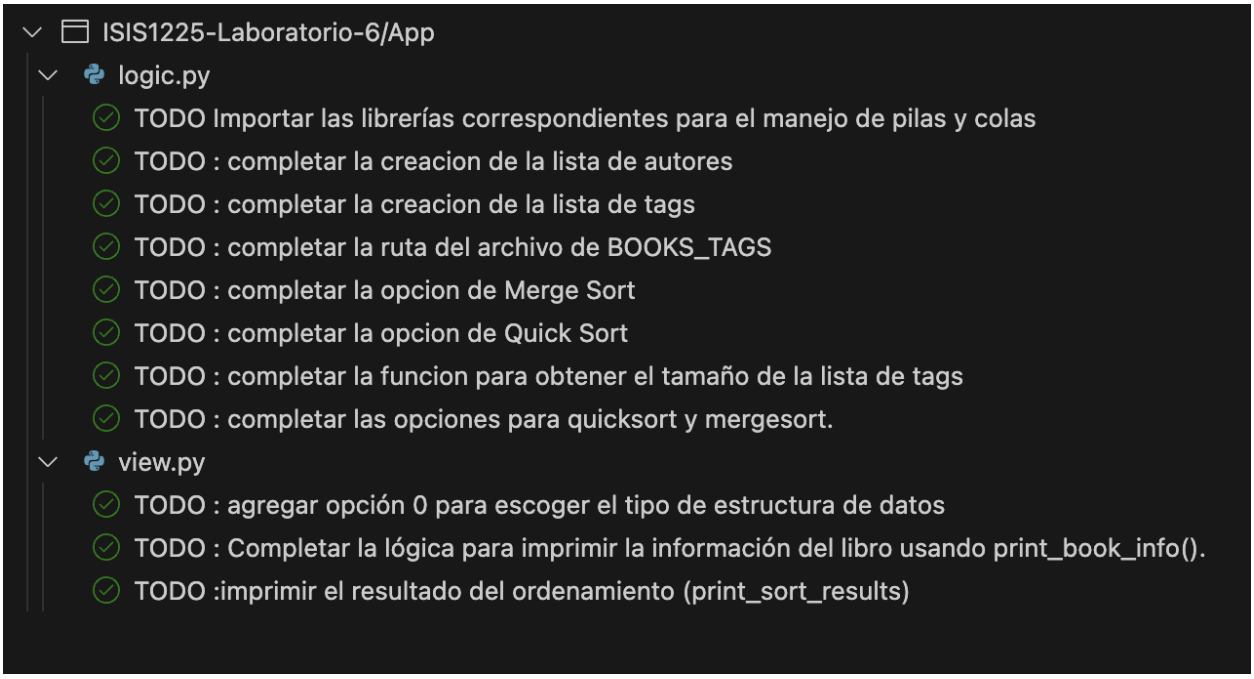

Ilustración 9 TODO

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:

Cada estudiante debe ejecutar todas las pruebas de rendimiento propuestas en este aparte.

- Diligencia la Tabla 1 con la información de los computadores donde cada estudiante ejecutará las pruebas de los algoritmos de ordenamiento iterativo ubicado en el documento **Observaciones-lab5.docx**.

	Máquina 1	**Máquina 2**	**Máquina 3**
Procesadores			
Memoria RAM (GB)			
Sistema Operativo			

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

Por ejemplo:

	Máquina 1	**Máquina 2**	**Máquina 3**
Procesadores	Intel(R) Core (TM) i7-55000 CPU @2.40GHz 2.40GHZ	Intel(R) Core (TM) i7-10750H CPU @ 2.60GHz 2.59 GHz	Intel(R) Core (TM) i2400H CPU @ 2.2GHz 2.60 GHz
Memoria RAM (GB)	16.0 GBGB	32.0 GB	8.0 GB

Sistema Operativo	Windows 10 Pro - 64 bits	Windows 10 Pro - 64 bits	Windows 10 Pro – 64 bits

Tabla 2. ejemplo especificaciones de las máquinas para ejecutar las pruebas de rendimiento.

- Ejecute todas las pruebas de rendimiento propuestas en la misma máquina que reporto.
- 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 ordenamiento **demora 15 minutos o más debe de dar por terminada la prueba** y continuar con las pruebas de otro algoritmo de ordenamiento.

En esta fase del laboratorio, se evaluará el rendimiento de los algoritmos de ordenamiento **Merge Sort** y **Quick Sort** 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 recursivos

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

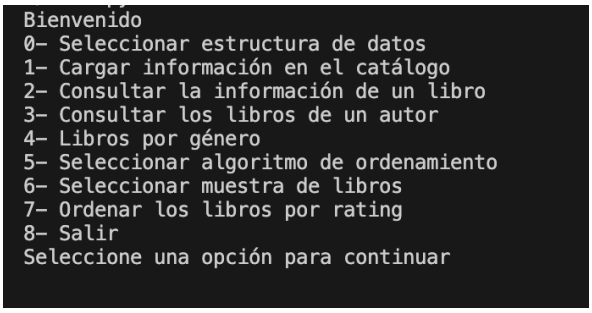

Ilustración 9 Menú de opciones

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

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)**
    - Aunque se le permite escoger cualquier algoritmo de ordenamiento visto para este momento solo elija entre **Merge Sort** o **Quick Sort.**
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 seguir la tabla y dar la cantidad de datos y no el porcentaje es decir si quiero el 10% de los datos y mis datos son 1000 le ingreso 100
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.

2.6 Comparar Algoritmos y Estructuras

Después de registrar los datos de los algoritmos de ordenamiento recursivos, cada integrante del grupo deberá graficar los resultados para comparar el comportamiento de los algoritmos y responder preguntas de análisis. Para ello, deberán utilizar el archivo "ISIS1225 – Tablas de Datos Lab 6.xlsx" para registrar la información.

Cada estudiante, de manera individual en Excel, deberá cumplir con las siguientes instrucciones:

- Completar dos (2) gráficas comparativas para cada algoritmo de ordenamiento recursivo.
- Calcular **las líneas de tendencia más apropiadas** para los datos en cada una de las gráficas.
- Comparar los resultados obtenidos en las máquinas de trabajo del grupo con la **complejidad teórica** de los algoritmos.

Importante: Por ahora, **no es necesario llenar la tabla "TIEMPOS DE EJECUCIÓN MEJORES ALGORITMOS ARRAYLIST [ms]"**. Del mismo modo, **ignore la pestaña de la gráfica denominada "bests"** hasta que se le indique lo contrario.

2.7 Comparar tiempos de ejecución de los mejores algoritmos

Discuta con sus compañeros y analicen cuál es el algoritmo que presenta mejor eficiencia. Recuerde lo trabajado en la práctica de laboratorio anterior, donde debió identificar el mejor algoritmo de ordenamiento iterativo según su criterio.

Ahora, complete **cada una de las tablas** en la sección **"Comparación de tiempos de los mejores algoritmos"** del documento de observaciones, siguiendo los mismos pasos utilizados para los algoritmos recursivos. Sin embargo, en el **paso 5**, deberá seleccionar:

1. **El mejor algoritmo recursivo** según su criterio.
2. **El mejor algoritmo iterativo** según su desempeño en la práctica anterior.

Complete la tabla correspondiente para **cada tamaño de muestra**.

Nota: Recuerde que ya debió registrar estos datos, así que puede copiar y pegar los valores obtenidos en el laboratorio anterior y los datos obtenidos en esta práctica.

Después de registrar los datos en el documentó de observaciones compartido, diríjase al archivo Excel individual y registre los datos en la sección correspondiente. Finalmente, analice la gráfica obtenida para evaluar los resultados.

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

2.8 Documentar el análisis de los ordenamientos

Al completar las pruebas, y responder las preguntas, verifique que los resultados de cada uno de los miembros del grupo se encuentren compilados en el archivo llamado **Observaciones-lab 6.docx**.

El documento debe incluir lo siguiente:

a. Nombres, apellidos, código de estudiante de los integrantes del grupo.
b. Cada estudiante incluye los datos reportados en cada una de las tablas.
c. El grupo incluye las respuestas a las preguntas de análisis.

Es importante que cada estudiante realice las pruebas de forma individual, por lo que se espera que al final del laboratorio se presenten 3 archivos de Excel (o 2, en caso de grupos de dos estudiantes) y un documento en formato PDF, correspondiente al archivo "observaciones-lab6".

OJO: Aunque se proporciona el formato del documento de observaciones en Word, es necesario que conviertan y entreguen el archivo final en PDF. Recuerde responder estas preguntas utilizando el archivo "large".

Para garantizar la correcta entrega de los archivos, cada grupo debe guardar y añadir los documentos correspondientes en la carpeta Docs del repositorio. Se espera que, al finalizar el laboratorio, la carpeta contenga tres archivos de Excel con los datos reportados por cada integrante, 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. Es fundamental que el archivo de observaciones sea convertido a PDF antes de ser subido.

A continuación, se presenta un paso a paso detallado para realizar la subida de los documentos correctamente. Este procedimiento asume que cada estudiante está trabajando en una rama diferente,

lo que significa que cada integrante del grupo debe gestionar sus propios cambios sin interferir con los archivos de los demás.

El paso a paso se ejemplifica utilizando el estudiante **Est1**, pero aplica de la misma manera para los tres integrantes del grupo. La única diferencia radica en los nombres de los archivos, los cuales deben finalizar con el número correspondiente a cada estudiante (*Est1, Est2, Est3*). Siguiendo esta metodología, se garantiza una correcta organización de los archivos en el repositorio y se evita la sobreescritura accidental de información.

Nota: Si no están utilizando ramas, este proceso se realiza una única vez en la rama main y lo hará una sola persona del grupo quien debe subir todos los archivos, recuerden que los demás integrantes solo necesitan actualizar sus repositorios para sincronizar los cambios.

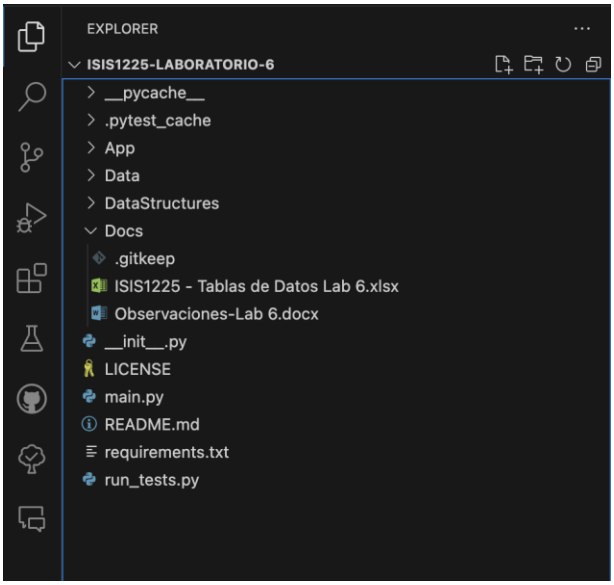

Ilustración 101 Carpeta Docs sin cambios

La ilustración 11 muestra el estado inicial de la carpeta, donde solo se encontraba un archivo de Excel (*ISIS1225 - Tablas de Datos Lab 6.xlsx*) y un documento en formato Word (*Observaciones-Lab 6.docx*).

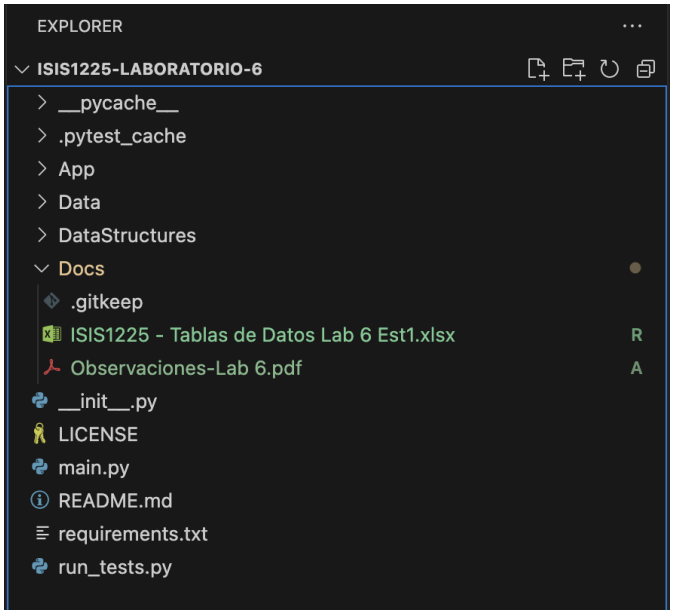

Ilustración 11 Carpeta docs con los archivos correspondientes Est1

en la ilustración 12, se observa la carpeta **Docs** después de que el estudiante ha agregado sus archivos individuales. Ahora se incluyen **el archivo de Excel** (ISIS1225 - Tablas de Datos Lab 6 Est1), y el documento de observaciones que ha sido convertido a **PDF** (*Observaciones-Lab 6.pdf*), como se solicitó en las instrucciones del laboratorio.

NOTA: Se sugiere que solo un estudiante del grupo agregue el PDF a la carpeta.

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.

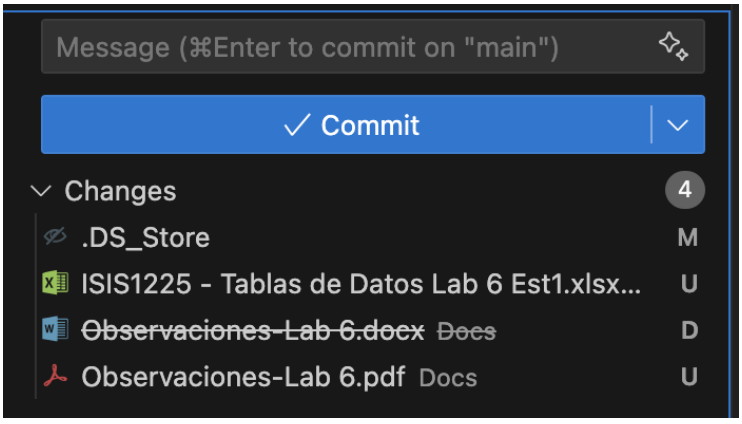

Ilustración 12 Archivos sección changes

En la ilustración 13, se pueden ver los archivos en la sección *Changes*. Esto significa que han sido detectados como nuevos o modificados, pero aún no han sido añadidos al repositorio. En este punto, cada estudiante debe asegurarse de que los archivos correctos estén listos para ser agregados. Recuerde se recomienda que un solo integrante del grupo debe encargarse de subir el documento de observaciones en formato PDF.

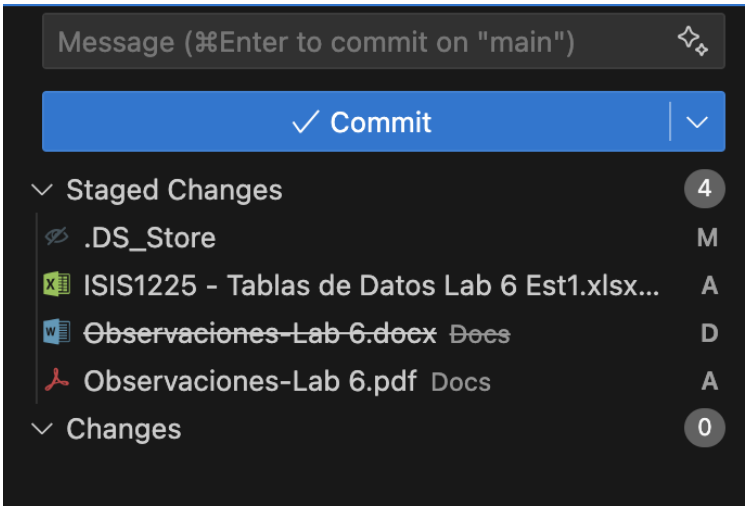

Ilustración 13 Staged Changes

En la **Ilustración 14**, los archivos han sido movidos a la sección **Staged Changes**, lo que indica que han sido añadidos al **staged** y están listos para ser confirmados (**commit**). Antes de proceder, es fundamental verificar que solo los archivos correspondientes estén incluidos para evitar conflictos en el repositorio.

Una vez que los archivos estén en **Staged Changes**, cada estudiante debe realizar el commit, asegurándose de agregar un mensaje claro, por ejemplo: **"Archivos docs Est1"**, reemplazando Est1 por su identificación correspondiente (Est2 o Est3, según aplique). Si están trabajando en la rama main, omitan la identificación y realicen el commit únicamente con el mensaje: **"Archivos docs**".

Posteriormente, deben subir los cambios al repositorio remoto realizando un push.

Por último, es importante hacer **merge** con la rama **main**, esto lo aprendió en el laboratorio 2 recuerde coordinarse con sus compañeros para que los cambios se suban de manera ordenada y evitar conflictos en el repositorio. Recuerden que, si un compañero ya subió cambios, es necesario **actualizar la rama main** antes de que usted inicie a hacer el merge, asegurándose así de trabajar con la versión más reciente del proyecto.

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 tres 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.