LABORATORIO 4: PILAS Y COLAS

1 Objetivo

Comprender la implementación de los Tipos Abstractos de Datos **Pila (Stack) y Cola (Queue)** y su uso en la solución de problemas. Al finalizar este laboratorio el estudiante estará en capacidad de:

1) Desarrollar las operaciones principales de los TAD **Pila y Cola**.
2) Utilizar las estructuras de datos **Pila y Cola** para almacenar y gestionar datos de manera eficiente.
3) Realizar la carga de archivos de tipo **`.csv.`**
4) **Medir y analizar el desempeño** de las operaciones fundamentales en pilas y colas, tomando tiempos de ejecución para evaluar su eficiencia.
5) Comparar las implementaciones con arrays (ArrayList) y listas enlazadas (LinkedList), identificando en qué casos cada una es más eficiente en términos de tiempo y espacio.

2 Trabajo Propuesto

El trabajo del laboratorio se basa en el proyecto Laboratorio-4. Bifurque el repositorio en su organización siguiendo las convenciones de nombramiento **Laboratorio4-G<<YY>>** donde **YY** el número del grupo de trabajo (ej.: Laboratorio4-G01).

Agregue los nombres y usuarios de los integrantes del equipo en el archivo README del repositorio, esto solo debe hacerlo un estudiante del equipo. No olvide subir los cambios a GitHub y actualizar los repositorios de los demás estudiantes del grupo.

IMPORTANTE: Descargue o copie los datos de prueba GoodReads usados anterior mente en los laboratorios 2 y 3 desde el aula unificada de BN y guárdelos en la carperta /Data/GoodReads/ del proyecto.

2.1 Implementación de estructuras

En este laboratorio, tal como lo hicieron con la estructura de listas, implementarán nuevas estructuras de datos. Usarán la documentación del curso, encontrada en DISC – Data Structures como guía para el desarrollo de las estructuras que le pediremos en el curso.

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 Página de inicio de documentación

Ya en los laboratorios de ArrayList y SingleLinkedList, 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 **pilas y colas**, por lo que deberán ingresar a la documentación correspondiente, como se muestra en la siguiente captura de pantalla:

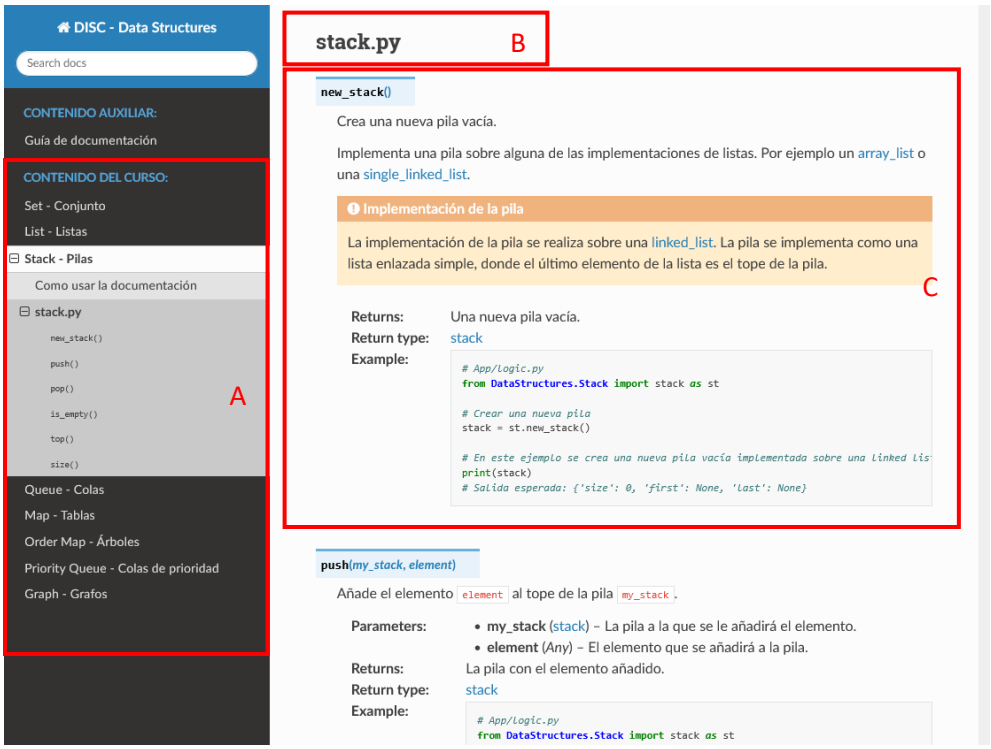


Ilustración 2. Implementación de la pila

En la ilustración 2 se pueden identificar los 3 apartados de la documentación con los cuales se interactuará para identificar los desarrollos de las estructuras.

A. **Listado de estructuras e implementaciones:** Este apartado permite navegar entre las diferentes estructuras de datos vistas en el curso, junto con sus implementaciones y los métodos de cada una. En este laboratorio, trabajarán con pilas (Stack) y colas (Queue), por lo que deberán revisar las respectivas secciones en la documentación.
B. **Archivo de la implementación de estructura:** Cada estructura tiene un archivo 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 los siguientes archivos dentro de la carpeta DataStructures:

    - stack.py para la implementación de pilas, en la carpeta Stack
    - queue.py para la implementación de colas, en la carpeta Queue

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 cada descripción para garantizar el correcto funcionamiento de su código.

2.2 Implementación de funcionalidad

Una vez clonado el repositorio en su máquina local, encontrará una estructura de proyecto similar a la presentada en la imagen. La organización del proyecto sigue el mismo esquema trabajado en el Laboratorio 3, por lo que la carpeta DataStructures debe contener la carpeta List, en la cual se deben incluir las implementaciones de array_list y linked_list desarrolladas en el laboratorio anterior. Para este laboratorio, además de la carpeta List, se deberán agregar las carpetas Stack y Queue, cada una con una estructura similar, pero enfocadas en la implementación de pilas y colas, respectivamente. Estas nuevas carpetas contendrán únicamente los archivos de prueba (Tests), mientras que la implementación deberá ser desarrollada por el equipo.

A partir de este momento, el objetivo es que, como equipo, continúen desarrollando las estructuras de datos vistas en el curso, consolidando su implementación para utilizarlas posteriormente en los retos.

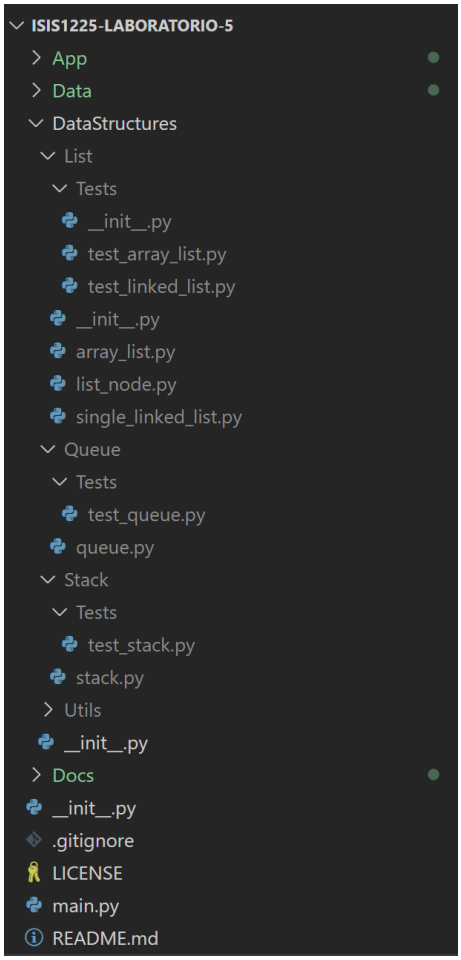


Ilustración 3 - Estructura del laboratorio 3

2.2.1 Implementando el TAD Cola

Teniendo en cuenta la documentación de DISC – Data Structures creará las primeras implementaciones de estructura de datos.

1. Se debe crear el archivo queue.py dentro de la carpeta Queue, respetando la estructura de archivos del laboratorio

2. Lea la documentación de la función *new_queue()* e identifique que se espera que se haga al momento de crear una nueva cola. Como resultado, se debe crear una cola vacía.
3. En *logic.py* importe la estructura recién creada de la siguiente manera:

```
from DataStructures.Queue import queue as q
```

4. De esta manera se importarán las estructuras que crearemos en los laboratorios del curso. Para comprobar que el desarrollo de una función esté correctamente hecho, hemos creado una serie de pruebas pensadas para comprobar el desarrollo mínimo de la estructura esperada. Es necesario aclarar que estas pruebas no validan la complejidad temporal del código implementado sino solamente validan el correcto nombramiento y resultados parciales de las funciones de la estructura, por lo tanto, se podría hacer el desarrollo de una función con complejidad exponencial que soluciona el problema, pero no es óptima.
5. Dado que en el laboratorio anterior ya trabajaron con pruebas unitarias y conocen la estructura de los archivos de pruebas, en este laboratorio seguirán el mismo enfoque. Abra el archivo test_queue.py ubicado en Datastructures/Queue/Tests y el archivo test_stack.py ubicado en Datastructures/Stack/Tests. Estos archivos contienen todas las pruebas unitarias para las estructuras cola y pila. A continuación, se muestra un ejemplo de estas pruebas unitarias.

```python

from DataStructures.Queue import queue as q
from DataStructures.Utils.utils import handle_not_implemented


Tabnine | Edit | Test | Explain | Document
def setup_queue():
    # Inicializa una cola vacía para pruebas
    return q.new_queue()


Tabnine | Edit | Test | Explain | Document
@handle_not_implemented
def test_new_queue():
    # Verifica que la cola se crea correctamente y está vacía
    my_queue = setup_queue()

    assert type(my_queue) == dict
    assert my_queue["size"] == 0
    assert my_queue["elements"] == []


Tabnine | Edit | Test | Explain | Document
@handle_not_implemented
def test_enqueue():
    # Verifica que los elementos se agregan correctamente al final de la cola
    my_queue = setup_queue()

    q.enqueue(my_queue, 1)

    assert my_queue["size"] == 1
    assert my_queue["elements"][-1] == 1


Tabnine | Edit | Test | Explain | Document
@handle_not_implemented
def test_dequeue():
    # Verifica que el `dequeue` retira y devuelve el primer elemento de la cola
    my_queue = setup_queue()
```

Ilustración 6 - Test unitarios en queue.py

6. Para facilitar la ejecución de los tests se ha creado un *script* llamado run_tests.py. Para correr el script ejecute el siguiente comando desde el directorio principal del laboratorio:

```
ISIS1225-Laboratorio-3> python .\run_tests.py
```

7. Al ejecutar el script aparecerá el menú de ejecución de los tests, ingrese la opción “3” para colas y “4” para pilas luego presione enter. Deberá aparecer una ejecución como la presentada a continuación:

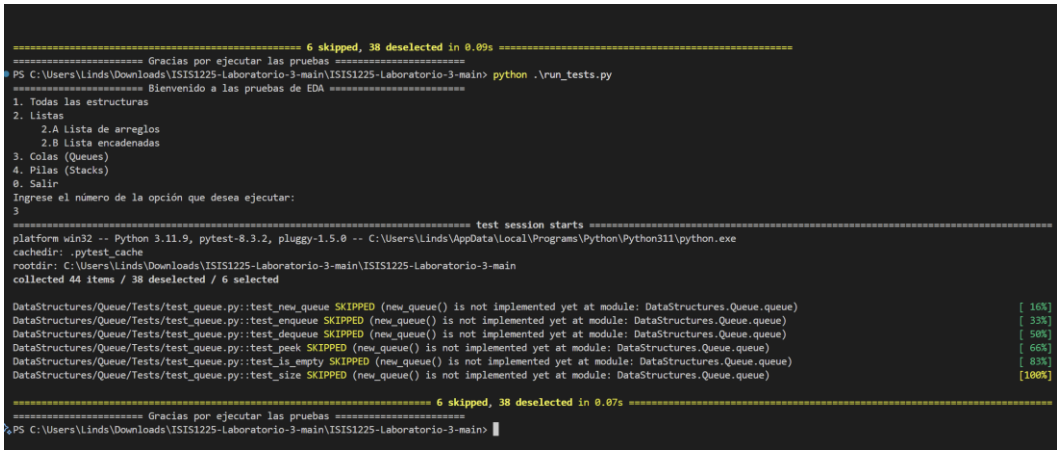


Ilustración 7 - Ejecución de tests sobre colas

8. Implemente las funciones del TAD Cola enqueue(), dequeue(), peek(), is_empty() y size().
9. Verifique su implementación realizando la ejecución de las pruebas y haga las correcciones que sean necesarias.
10. Guarde y suba los cambios de clase a Github. Cree un ***release*** con el título “Entrega Inicial Laboratorio 4” y el tag “0.0.1” en su repositorio grupal.

2.2.2 Implementando el TAD pila y la aplicación

Continue trabajando en la implementación de las funciones necesarias para el desarrollo de la estructura *stack* tomando como guía la documentación del curso y la ejecución de las pruebas compartidas en este laboratorio

1. Se debe crear el archivo stack.py dentro de la carpeta Stack, respetando la estructura de archivos del laboratorio
2. Lea la documentación de la función *new_stack()* e identifique qué se espera como resultado al momento de crear una nueva pila.
3. En *logic.py* importe la estructura de la siguiente manera:

```
from DataStructures.Stack import stack as st
```

4. Implemente las funciones del TAD Pila new_stack(), push(), pop(), is_empty(), top() y size().
5. Complete las funciones que se encuentran incompletas en el archivo logic.py como se presenta en la **¡Error! No se encuentra el origen de la referencia.** para que la aplicación quede completamente implementada.

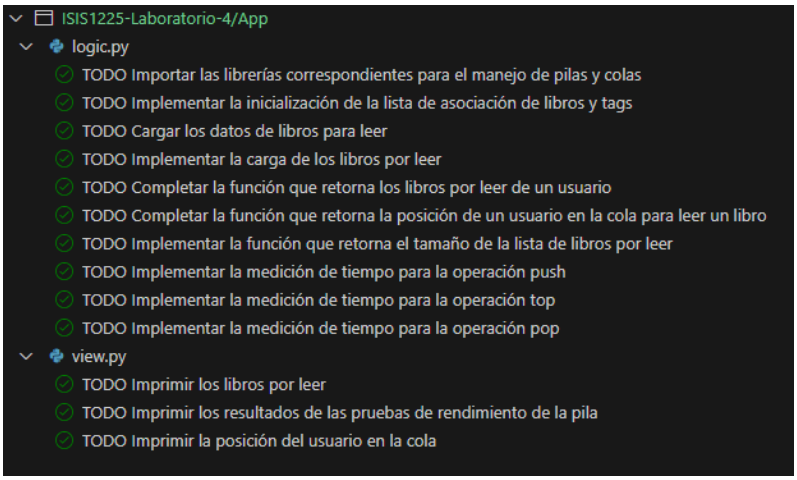


Ilustración 9 - ToDos del laboratorio 4

6. Asegúrese que al finalizar la implementación de las estructuras esté correctamente implementada ejecutando las pruebas unitarias y que estas estén completamente en color verde tal cómo se presenta en la **¡Error! No se encuentra el origen de la referencia.**

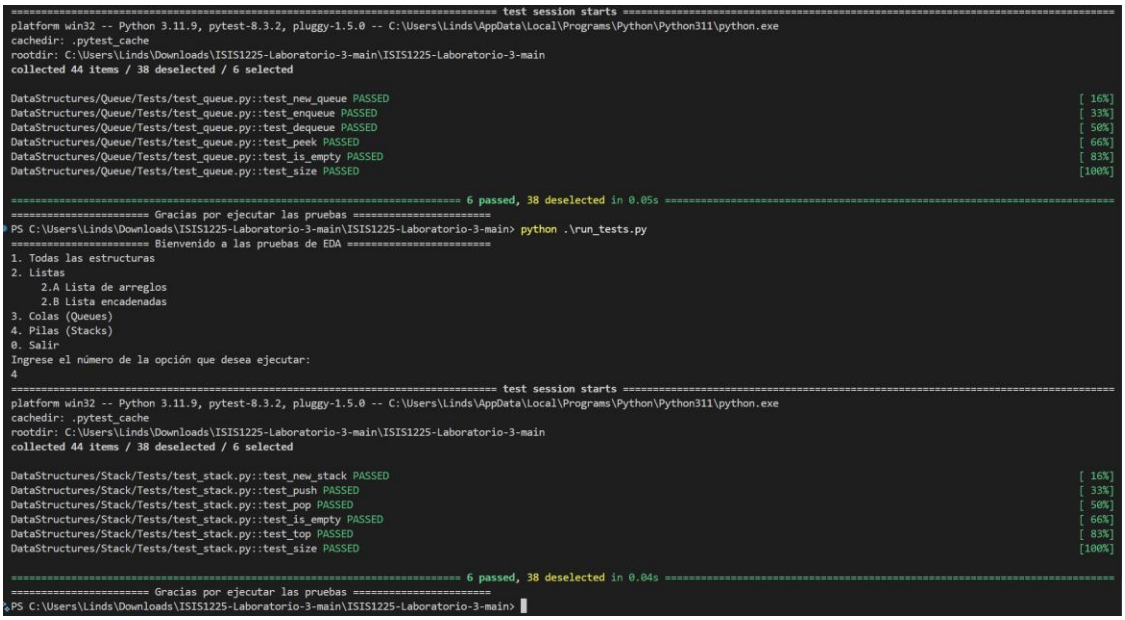


Ilustración 10 - Ejecución final de las pruebas para Queue y Stack

2.2.3 Pruebas de tiempo de ejecución

Las funciones get_time() y delta_time(start, end) se encuentran implementadas en el archivo *logic.py* y se utilizan para medir el tiempo de ejecución de un algoritmo en milisegundos.

```python
def get_time():
    """
    devuelve el instante tiempo de procesamiento en milisegundos
    """
    return float(time.perf_counter()*1000)


def delta_time(start, end):
    """
    devuelve la diferencia entre tiempos de procesamiento muestreados
    """
    elapsed = float(end - start)
    return elapsed
```

get_time() obtiene el instante actual de procesamiento, mientras que delta_time() calcula la diferencia entre dos instantes para determinar el tiempo transcurrido.

En el archivo *view.py*, se utiliza el siguiente print para mostrar el tiempo de ejecución formateado con tres decimales y acompañado de la etiqueta "[ms]", permitiendo al usuario visualizar el rendimiento del algoritmo.

```python
print("Tiempo de ejecución para enqueue:",
f"{queue_results['enqueue_time']:.3f}", "[ms]")
```

Para este laboratorio, además de implementar los TAD de pila y cola, deberán entregar un informe y un archivo de Excel con los datos de rendimiento obtenidos. La plantilla para estos informes se encuentra en la carpeta "Docs" del proyecto.

Para realizar las pruebas, se usa la opción del menú "**4- Ejecutar pruebas de rendimiento**". Al seleccionar esta opción, primero se debe indicar el número de registros que se usarán durante las pruebas y posteriormente se mostrarán los tiempos de ejecución. Deben variar el tamaño de la muestra de acuerdo con los valores que encontrará en la plantilla y anotar los tiempos obtenidos en las casillas correspondientes. Para la realización de las pruebas, es necesario usar el archivo **books.csv**, el cual contiene suficientes registros para la ejecución de todas las pruebas.

Tengan en cuenta que en los archivos del informe se registran los tiempos de ejecución para las estructuras de pila y cola, utilizando ambas implementaciones de lista: **Array List** y **Single Linked List**. Por lo tanto, deberán ejecutar las pruebas dos veces, una con cada implementación.

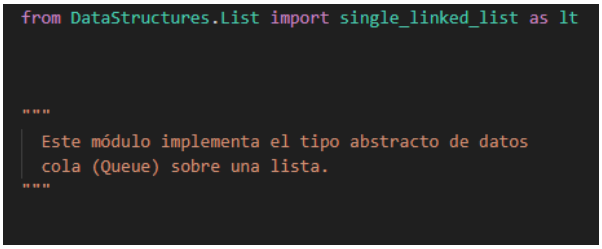
Para alternar entre una implementación y otra, solo es necesario modificar la declaración de **import** al inicio de los archivos donde se implementan los TAD. Por ejemplo, si implementaron el TAD Cola con Array List de manera similar a como se muestra en la imagen:

```python
from DataStructures.List import array_list as lt



"""
  Este módulo implementa el tipo abstracto de datos
  cola (Queue) sobre una lista.
"""
```

Deben cambiar a la implementación con Single Linked List, de manera similar a como se ve a continuación:



Deben hacer el mismo cambio en el archivo de la Pila para poder realizar la totalidad de las pruebas.

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-lab4".

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".

3 Entrega

3.1 Confirmar cambios finales

Confirme los cambios finales siguiendo los pasos aprendidos en prácticas anteriores y cree un *release* con el título “Entrega Final Laboratorio 4” y el tag 1.0.0 antes de la fecha límite de entrega del laboratorio

3.2 Compartir los resultados con los evaluadores

Finalmente, para realizar la entrega de los resultados de la práctica de laboratorio se debe enviar el enlace (URL) del repositorio GitHub a los monitores y profesores de **su sección**, antes de la fecha definida en la actividad del curso.