**RESUMEN LABORATORIO NO. 2: LISTAS Y ORDENAMIENTOS BÁSICOS**

# Objetivos

Crear una implementación de estructura de datos tipo lista.

1. Implementar algoritmos de ordenamiento básicos.
2. Evaluar los tiempos de ejecución de los algoritmos implementados.
3. Desarrollar las pruebas para validar las operaciones sobre la estructura de datos.
4. Utilizar adecuadamente el ambiente de trabajo distribuido configurado.

# Metodología

La dinámica se divide en tres etapas con diferentes recursos pedagógicos. La primera etapa es el trabajo preparatorio de carácter principalmente individual. Segundo, en el encuentro sincrónico se aclararán dudas y veremos el estado de la práctica. Y tercero, la retrospectiva se entregarán los productos de la práctica para ser evaluados por los monitores y profesores.

# Desarrollo

El laboratorio dos (2) tienen dos (2) temas. Primero, crear un TAD lista (ADT List) utilizando la implementación del curso. Y segundo, utilizar algoritmos de ordenamiento (Sorting) básicos para ordenar elementos de una lista.

El resumen de las actividades y temas puede verse en la Ilustración 1, mientras que el material y recursos asociados a las actividades pueden consultarse en la sección 4 de Materiales & Recursos.

Las tareas específicas de esta práctica son:

* Estudiar el código del laboratorio No. 2 (ver sección 3.1).
* Completar el examen técnico del laboratorio no. 2 (ver sección 3.2).
* Implementar las pruebas en ADT List (ver sección 3.3).
* Modificar los ordenamientos básicos (ver sección 3.4).
* Integrar las implementaciones de listas y ordenamientos (ver sección 3.5).
* Analizar los resultados de listas y ordenamientos (ver sección 3.6).
* Compartir el producto final con los evaluadores (ver sección 3.7).

Por último, recuerde que lo aprendido en el laboratorio No. 1 servirá para configurar y desarrollar el trabajo de este.

## Estudiar el código del laboratorio No. 2:

Para familiarizarse con la implementación del código del laboratorio dos, lea el código y documentación de las siguientes rutas:

* La implementación de lista en la dirección relativa *\*\Lab2\_202020\ADT\list.py*
* La implementación del arreglo y las listas encadenadas en las siguientes direcciones relativas:
  + Código en *\*\Lab2\_202020\DataStructures\arraylist.py*
  + Código en *\*\Lab2\_202020\DataStructures\singlelinkedlist.py*
  + Código en *\*\Lab2\_202020\DataStructures\listnode.py*
* La implementación de los algoritmos de ordenamiento en las siguientes direcciones relativas:
  + Código en *\*\Lab2\_202020\Sorting\selectionsort.py*
  + Código en *\*\Lab2\_202020\Sorting\insertionsort.py*
  + Código en *\*\Lab2\_202020\Sorting\selectionsort.py*

## Completar el examen técnico del laboratorio No. 2:

Para responder al cuestionario técnico de los temas del laboratorio No. 2, diríjase a la página de evaluaciones de su sección en Sicuaplus.

## Implementar las pruebas en ADT List:

Después de familiarizarse con la estructura de ADT List y sus implementaciones de lista enlazada (Singlelinkedlist) y arreglo (Arraylist) implemente las pruebas Pytest con las siguientes instrucciones:

* Encontrar la carpeta de pruebas del código en la dirección relativa *\*\Lab2\_202020\Test\\*.*
* Crear una carpeta *lab2* con los archivos de prueba *test\_arraylist.py* y *test\_singlelinkedlist.py.*
* Completar las pruebas de listas utilizando los archivos de películas y elenco.

## Modificar los ordenamientos básicos:

Después de familiarizarse con los algoritmos implementados de ordenamiento, siga los siguientes pasos para implementar las pruebas Pytest de *SelectionSort*, *InsertionSort* y *ShellSort.*

* Modificar el código de los ordenamientos para que funcionen con los campos del archivo del reto No. 1 y pueda cumplir con el requerimiento 2 y 3; para esto debe modificar la función ***less*** que se envía por parámetro al algoritmo.
* Crear una función para comparar llamada ***greater*** para que realice un ordenamiento descendente al ser utilizada como parámetro en los algoritmos.
* Encontrar la carpeta de pruebas del código en la dirección relativa *\*\Lab2\_202020\Test\\*.*
* En la carpeta *lab2* crear los archivos de prueba *test\_selectionsort.py*, *test\_insertionsort.py* y *test\_selectionsort.py.*
* Completar las pruebas de ordenamiento para que se puedan utilizar los archivos de películas y elenco.

## Integrar las implementaciones de Listas y Ordenamientos

Después de completar las pruebas se debe extender la aplicación con los siguientes pasos:

* Extender el menú de la aplicación *(\*\Lab2\_202020\App\app.py)* y crear la opción 5 y modifique la función *orderElementsByCriteria*, la cual ordenará la lista utilizando las funciones de ordenamiento e imprimirá los tiempos de ejecución y cumplirá con el Req No 2. Del reto.
* Modificar *countElementsByCriteria* para cumplir con el requerimiento No. 3 del reto, de ser necesario puede adicionar más parámetros a la función.
* Comparar el desempeño de las implementaciones de Lista (arreglo y lista encadenada) al cargar los datos del reto registrando los tiempos promedios en la Tabla 1 utilizando las versiones \*-small.csv de los archivos.
* Comparar el desempeño de los tres tipos de ordenamientos contra las implementaciones de ADT List (arreglo y lista encadenada) al procesar los datos del reto registrando los tiempos promedios de ejecución en la Tabla 2 utilizando las versiones \*-small.csv de los archivos.

IMPORTANTE: Para obtener resultados consistentes para este análisis se recomienda ejecutar por lo menos 5 veces la prueba y promediar los resultados que de la Tabla 1. y Tabla 2.

|  |  |  |
| --- | --- | --- |
| Fuente de datos | Arraylist [ms] | Singlelinkedlist [ms] |
| Películas (MoviesDetailsCleaned) |  |  |
| Elenco (MoviesCastingRaw) |  |  |

* Tabla 1. Tabla de rendimiento para cargar archivos como lista enlazada y arreglo.

|  |  |  |
| --- | --- | --- |
| Algoritmo | Arraylist [ms] | Singlelinkedlist [ms] |
| SelectionSort |  |  |
| InsertionSort |  |  |
| ShellSort |  |  |

Tabla 2. Tabla de rendimiento para los distintos algoritmos de ordenamiento en listas enlazadas y arreglos.

## Analizar los resultados de listas y ordenamientos básicos:

Con los resultados de la Tabla 1. y Tabla 2 reflexione y responda las siguientes preguntas con respecto a las estructuras y algoritmos.

Con los resultados de la Tabla 1. y Tabla 2 reflexione y responda las siguientes preguntas con respecto a las estructuras y algoritmos.

* ¿Qué diferencias se observan en desempeño de la carga de datos entre arreglo (*Arraylist*) y lista sencillamente encadenada (*Singlelinkedlist*)?

Es más rápido cargar datos desde un tipo arraylist pues en este tipo de ordenamiento se tiene acceso a todos los elementos del arreglo por medio de las referencias indexadas en memoria. Llegar al elemento n de la lista tiene un ordenamiento temporal constante O(1). Por el contrario, una linked list depende de recorrer la lista entera para llegar a un elemento que se encuentre en N posición pues los referentes dependen del elemento inmediatamente anterior, en este caso el ordenamiento de tiempo para cargar un elemento de la lista es O(N)

* ¿Cuál de las dos implementaciones (*Arraylist* y *Singlelinkedlist*) tiene mejor desempeño? Y ¿Por qué?

No se puede afirmar que un tipo de ordenamiento sea mejor en términos absolutos. El desempeño depende del tipo de tarea a realizar. En el punto anterior ya se discutió de la carga de un elemento que pertenece al ordenamiento y como este difiere entre los dos tipos. Si, por ejemplo, realizáramos una inserción en una posición n especifica esta seria más rápida en una linked list que en un array

* ¿Qué diferencias existen entre cargar los archivos de películas (*MoviesDetailsCleaned*) y elenco (*MoviesCastingRaw*)?, ¿Por qué se presentan estas diferencias?

A pesar de que los archivos tienen el mismo numero de filas tienen un numero de columnas diferentes. Los archivos de details tienen un volumen de información mayor. La información contenida en estas columnas Tambien ocupa mayor tamaño en los archivos de details que en los de casting.

* ¿Qué diferencias en el desempeño se observan entre los tres algoritmos de ordenamiento?

Los algoritmos de ordenamiento tienen un desempeño diferente al realizar una misma tarea. El Quicksort es el tipo de algoritmo más rápido para las tares realizadas. Los insertion sort y mergesort son considerablemente mas lentos pues estos tienen métodos de recorrer y organizar las estructuras de datos que son menos eficientes.

## Compartir el producto final con los evaluadores:

Para entregar exitosamente sus resultados de este laboratorio, por favor recuerde las siguientes indicaciones:

* Invitar al profesor y monitores del laboratorio asignados.
* Incluir en el **README** del repositorio los datos completos de los integrantes del grupo (nombre completo, correo Uniandes y código de estudiante).
* Incluir la Tabla 1. y la Tabla 2 diligenciadas en un documento.
* Responder las preguntas de análisis en el documento Word (ver sección 3.6).
* Incluir en el documento los datos completos de los integrantes del grupo (nombre completo, correo Uniandes y código de estudiante).
* Incluir el documento en formato Word o PDF en el repositorio GIT.
* Entregar el enlace (URL) del repositorio por Sicuaplus.

Recuerden que cualquier documento solicitado durante las actividades debe incluirse en el repositorio GIT y que solo se calificara hasta el último **COMMIT** realizado dentro de las fechas límites.

# Materiales & Recursos

Ilustración 1. Diagrama de actividades y temas generales para el laboratorio No 2.

## De Preparación:

|  |  |  |
| --- | --- | --- |
| ACTIVIDAD PEDAGÓGICA | TEMA TRATADO | MATERIAL ASOCIADO |
| Copiar/Fork del repositorio para Lab No. 2 | ADT List | * Repositorio GIT Laboratorio No. 2, URL: <https://github.com/ISIS1225DEVS/Lab2_202020> |
| Estudiar la implementación de ADT List y Datastructures | ADT List | * Repositorio GIT Laboratorio No. 2. * Instrucciones para estudiar el código del laboratorio No. 2 (ver sección 3.1) * Video Uniandes de implementación de List TAD (ARRAY\_LIST), URL: <https://web.microsoftstream.com/video/f22f2ecf-b028-4b9d-8f26-561532b1c6a1> * Video Uniandes de implementación List TAD (LINKED\_LIST), URL: <https://web.microsoftstream.com/video/10d56408-2ef8-41f6-ba26-06308a4abf2c> * Video de como hacer un recorrido en una lista Python y en ADT List, URL: <https://web.microsoftstream.com/video/73ac7cd7-6f31-4a27-9174-18fbdba27ed2> * Recursos y material adicionales:   + Abstract Data Type (ADT) In Python, URL: <https://medium.com/@tssovi/abstract-data-type-adt-in-python-33e6ce1f961e>   + Python Data Structures Tutorial, URL: <https://www.datacamp.com/community/tutorials/data-structures-python>   + Introduction to Linked Lists (Data Structures & Algorithms #5), URL: <https://www.youtube.com/watch?v=WwfhLC16bis&list=PLBZBJbE_rGRV8D7XZ08LK6z-4zPoWzu5H&index=5>   + Python Data Structures #2: Linked List, URL: <https://www.youtube.com/watch?v=JlMyYuY1aXU> |
| Implementar pruebas ADT List | ADT List | * Archivos CSV del Reto No. 1. * Repositorio GIT Laboratorio No. 2. * Instrucciones para implementar las pruebas ADT List (ver sección 3.3) * Video Uniandes de Cómo hacer un test con Pytest + leer archivos CSV, URL: <https://web.microsoftstream.com/video/16c7a079-eb7f-4f48-a98b-df61e2c5047e> * Video Uniandes de como implementar comparadores en List ADT, URL: <https://web.microsoftstream.com/video/e3923626-566d-4411-a8f5-9b9cc30deb60> * Recursos y material adicionales:   + Pytest: helps you write better programs, URL: <https://docs.pytest.org/en/stable/>   + Pytest Course, URL: <https://www.youtube.com/playlist?list=PLJsmaNFr5mNqSeuNepT3IaMrgzRMm9lQR>   + Effective Python Testing With Pytest, URL: <https://realpython.com/pytest-python-testing/> |
| Integrar objetivos del Laboratorio No. 2 con el Reto No. 1. | Ordenamientos básicos | * Video Reto No. 1, URL: <https://web.microsoftstream.com/video/91eab175-1b5d-42de-a502-be8b7a711703> * Video Laboratorio No 2, URL: <https://web.microsoftstream.com/video/cc8d3460-534e-46a4-a096-24ddc5dad2ff> |
| Estudiar los ordenamientos de listas. | Ordenamientos básicos | * Repositorio GIT Laboratorio No. 2. * Instrucciones para estudiar el código del laboratorio No. 2 (ver sección 3.1) * Video Uniandes de cómo funciona el InsertionSort, URL: <https://web.microsoftstream.com/video/fd27d088-0a21-45d1-9422-e4b6ed5a1009> * Recursos y material adicionales:   + Insertion Sort Algorithm Explained (Full Code Included) - Python Algorithm Series for Beginners, URL: <https://www.youtube.com/watch?v=byHi41L9vTM>   + Selection Sort Algorithm Explained (Full Code Included) - Python Algorithms Series for Beginners, URL: <https://www.youtube.com/watch?v=4CykZVqBuCw>   + Quick Sort Algorithm Explained (Full Code Included) - Python Algorithm Series for Beginners, URL: <https://www.youtube.com/watch?v=kFeXwkgnQ9U> |
| Responder cuestionario sobre listas y ordenamientos. | Ordenamientos básicos | * Cuestionario con preguntas de conceptos básicos de ADT List y Ordenamientos básicos *(SelectionSort, InsertionSort, ShellSort).* **IMPORTANTE,** revisar los anuncios y actividades en las secciones oficiales del curso. (ver sección 3.2) |
| Iniciar modificaciones de los ordenamientos para el Req uno del Reto No. 1. | Ordenamientos básicos | * Repositorio GIT Laboratorio No. 2. * Instrucciones para modificar los ordenamientos básicos (ver sección 3.4). |

## En el Encuentro Sincrónico:

|  |  |  |
| --- | --- | --- |
| ACTIVIDAD PEDAGÓGICA | TEMA TRATADO | MATERIAL ASOCIADO |
| Completar las modificaciones de los ordenamientos. | Ordenamientos básicos | * Repositorio GIT Laboratorio No. 2. * Instrucciones para modificar los ordenamientos básicos (ver sección 3.4). |
| Analizar el comportamiento de pruebas de ordenamiento y ADT List | Ordenamientos básicos + ADT List | * Instrucciones para integrar las implementaciones y ordenamientos (ver sección 3.5) * Instrucciones para analizar el desempeño de los ordenamientos (ver sección 3.6). |
| Crear versión unificada del código en GIT | Ordenamientos básicos + ADT List | * Enlace GitHub con código actualizado en la rama Master |

## Para la Retrospectiva:

|  |  |  |
| --- | --- | --- |
| ACTIVIDAD PEDAGÓGICA | TEMA TRATADO | MATERIAL ASOCIADO |
| Crear informe de trabajo | ADT List+ Ordenamientos básicos | * Instrucciones para compartir el producto final con los evaluadores del Lab No. 2 (ver **Error! Reference source not found.**) * Enlace GitHub con código actualizado en la rama Master para Lab No. 2. * Invitación a los evaluadores (monitores y profesor) como colaboradores de la organización o proyecto. |