**Laboratorio 1 EDA grupo 1**

**Resumen**

El primer laboratorio del curso Estructuras De Datos Y Algoritmos se centró en el uso de Github y Visual Studio Code con el fin de crear el ambiente de trabajo necesario para la elaboración de los diferentes retos que se asignarán a lo largo del curso. Con base en el material disponible en Sicua (vídeos y documentos) se llevo a cabo el laboratorio que tuvo como entrega final el enlace del repositorio en Github en el cual podemos encontrar el código modificado y las respuestas a las preguntas planteadas en el laboratorio.

**Introducción:**

El siguiente laboratorio tiene como objetivo realizar el primer requerimiento del reto n°1 del curso de Estructuras De Datos Y Algoritmos (EDA), además aprender el uso de las funciones otorgadas en GitHub y VisualStudioCode para planear nuestro ambiente de trabajo el cual vamos a usar durante todo el semestre desarrollando software en equipo. En este laboratorio se siguieron los pasos plasmados en el documento de Word entregado desde los puntos 4.1.1 al 4.1.5 en los cuales se hace el proceso de creación de los grupos en GitHub hasta la creación de funciones dentro del repositorio otorgado por la clase, usando los datos de películas propuestos para el reto, que se encontraban en sicua plus.

**Respuesta a las preguntas:**

1. ¿Cómo implementaría la función? ¿Qué orden de complejidad tendría la función implementada?

La función se implementaría de la siguiente forma:

* Por medio de un ciclo buscar en el archivo de casting y guardar el id de todas las películas del director buscado (El id se guarda en una variable de tipo lista).
* Por medio de un ciclo buscar en el archivo de películas cada una de las películas del director buscado utilizando la lista que contiene la id de cada película.
* Al encontrar una película comparar su vote\_average con el valor 6 y si es mayor sumar el vote\_average en una variable que luego servirá para obtener el promedio y sumarle uno a una variable que se retornará como la cantidad de películas mejor valoradas del director buscado.
* Realizar el promedio dividiendo la variable que contiene la suma de los puntajes entre la variable que contiene la cantidad de películas mejor valoradas del director.
* Retornar la cantidad de películas mejor valoradas del director y el promedio obtenido.

La función implementada tendría orden de complejidad de 2n.

1. ¿Cuál es el ciclo regular para actualizar código en un repositorio GIT?

Lo primero es poner el repositorio en el IDE; luego se crea una rama en la cual se aplican las respectivas ediciones del código; posterior a eso, se hace un merge con la rama master y se verifica que no haya ningún tipo de error de compatibilidad, y finalmente con una pull request se actualiza el repositorio GIT.

1. ¿Qué ventajas y limitantes tiene el uso de Ramas/Branches?

Las branches pueden permitir el trabajo simultaneo de varios en un mismo programa, facilitando el trabajo de código en un ambiente remoto, sin embargo, para trabajar con las nuevas versiones del programa y las branches que pasaron a ser parte de la master, se requiere descargar otra vez el archivo, lo que puede llegar a ser molesto en algunos casos.

1. ¿Cuáles serían las buenas prácticas para solucionar conflictos?

Lo ideal serie revisar ambas partes del proyecto (el código previo al merge y posterior al merge) para de esa forma poder analizar que parte es la que está generando un conflicto con el código master, además, la documentación de los cambios puede ser de ayuda a la hora de revisar algún conflicto con el código.

1. ¿Qué orden de complejidad tendría las funciones (consulta y lectura de archivo)?

La función que se encarga de cargar el archivo para su uso posterior tendría una complejidad aproximada de O(n) ya que obligatoriamente deberá recorrer el archivo hasta su límite de n elementos. Mientras que la función de búsqueda de película por director puede llegar a tener una complejidad O(2n) en su peor caso, pues si el elemento de búsqueda estuviera localizado en la posición n, el algoritmo tendría de recorrer toda la selección de archivo hasta 2 veces (una para la búsqueda de director y otra para la búsqueda de las películas).

1. ¿Cómo podría reducir o aumentar la complejidad de la consulta?

Talvez la mejor forma de reducir la complejidad podría estar en las posiciones de los datos, pues estos están ubicados en el mismo lugar en ambos archivos. Una posible solución sería guardar la posición de las películas en una lista y luego usar esos valores para ir directamente en el segundo archivo, evitando así la revisión de la misma información 2 veces.

1. ¿Cómo afecta un TAD en la complejidad?, ¿Qué alternativas existen?

Un TAD permite generalizar a algún tipo de datos con amplia información (en nuestro caso, las películas, sus atributos y características) y transformarlo en algo mucho mas fácil de manejar (una agenda con el id de cada película y sus datos respectivos). Esto permitió que la elaboración de las funciones fuese mas sencilla, pues solo se debía buscar el id de una película y de esa forma hallar su director, su calificación, genero, etc.

Una alternativa podría ser el crear una clase llamada “película” y otra llamada “biblioteca”. La primera tendría la función de convertir cada línea del csv en una película, y la segunda las organizaría como una biblioteca en donde se puede tener acceso fácil solo buscando alguna categoría, ya sea el género, el director, los actores o el nombre. Todo esto creado por funciones para simplificar el código final.

**Conclusión**

A través del trabajo autónomo aprovechando el uso de videos, lecturas y guías tanto otorgadas por le curso como independientes se logró aprender los conceptos básicos del uso de GitHub y Visual Studio Code, sin embargo consideramos que aún hay vacíos en el proceso de aprendizaje dado a que la información guía fomentaba a la investigación total individual lo cual complicaba el proceso sin ninguna base de la cual apoyarse, sin embargo creemos que los objetivos se cumplieron en gran parte aunque podrían haber sido mejores.