**Entrega 01**

**Laboratorio 01**

**Grupo 04**

**Cuestionario**

**3.1.5**

a. ¿Cómo implementaría la función?

Para dar desarrollo a la función a implementar nos planteamos los siguientes factores: Tendríamos que añadir una segunda lista del archivo .csv la cual denominaríamos “list2”, después de esto, recorreríamos “list” buscando el nombre del director y, simultáneamente, guardando en una lista vacía el ID. Al final, se compararía, haciendo uso de el ID, si esa película tendría un “vote\_average” superior o igual a 6.

b. ¿Qué orden de complejidad tendría la función implementada?

El orden de complejidad de la función a implementar sería de N^2. Lo anterior debido a que en el peor caso, se estaría manejando dos ciclos anidados. El primer ciclo se implementa buscando si el ID guardado en la lista coincide con el archivo “.csv”. El segundo ciclo revisaría si el promedio es igual o superior a 6

**3.2.2**

1. ¿Cuál es el ciclo regular para actualizar código en un repositorio GIT?
2. Modificar el repositorio local.
3. Utilizar el comando “git add” para preparar la actualización
4. Subir el archivo actualizado a la rama/Branch propia.
5. Si al revisar el nuevo cambio, se esta completamente seguro de que el código es correcto se sube el repositorio actualizado a la rama principal o rama master
6. ¿Qué ventajas y limitantes tiene el uso de Ramas/Branches?

El uso de ramas/branches permite a los programadores desarrollar código de manera independiente, de un mismo repositorio. Gracias a lo anterior, si algún individuo comete un error, no estará alterando el repositorio principal permitiendo tener la rama principal intacta.

Un limitante al aplicar el uso de ramas en Git es que se pueden incurrir en conflictos, uno de estos puede ser que dos ramas del mismo repositorio al momento de hacer merge apunten a la modificación de una misma parte de código, por lo cual Git no puede decidir cual versión fusionar y cual no.

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

Al trabajar con ramas en Git es posible en incurrir en un conflicto, este se puede dar a consecuencia de diversos motivos. Al darse un conflicto entre dos ramas Git nos preguntará por cual de las dos nuevas versiones de código debería apuntar para fusionarlo con la rama master (hacer merge)

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

El orden de complejidad de las funciones es:

1.Lectura de archivo O(N).

2.Consulta N^2

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

Aumentar la consulta: Para aumentar la complejidad de la consulta podrá ponerse a revisar el archivo completo, porque no nos indicarían una columna exacta.

Disminuir la consulta: Para disminuir la complejidad de la consulta, en vez de darnos el número de la columna para consultar, que tuviéramos el nombre de la llave a acceder (vote\_average). Otra opción sería unificar las listas a consultar en un solo archivo .csv.

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

El TAD afecta mucho el tiempo de respuesta del algoritmo, lo anterior debido a que dependiendo de cómo este guardada la información en la memoria, será más rápida la respuesta. Existe diferentes alternativas como: arreglos, listas encadenadas, arboles binarios, tablas de hash, listas/matrices adyacencias