**Preguntas Laboratorio 1**

**EDA Sección 06 Equipo 4**

**Por: Julián David Sáenz, Andrés Felipe Lugo, Laura Vannesa Pardo**

1. **Compare los tiempos de carga con respecto al archivo.**

|  |  |
| --- | --- |
| Archivo | Tiempo (s) |
| test | 0.00 |
| MoviesCastingRaw-small | 0.824 |
| SmallMoviesDetailsCleaned | 0.824 |
| AllMoviesCastingRaw | 3.823 |
| AllMoviesDetailsCleaned | 4.624 |

En la tabla se registran los tiempos de carga de los archivos. En este caso es importante notar que el tiempo de carga depende de varios factores, el más importante es la cantidad de datos que se están procesando. En este caso es importante hacer la salvedad que esto obedece no sólo al número de filas el valor que se está contando en los métodos propuestos, sino también a la cantidad de columnas (campos) y la cantidad de información registrada en cada una (registros o número de filas en Excel). Esto se puede apreciar al comparar el tiempo de carga entre “Casting” y “Details”: a pesar de tener el mismo número de registros, no cargan en la misma en el mismo lapso, la diferencia entre los archivos básicamente son el número de campos como se observa en la siguiente tabla.



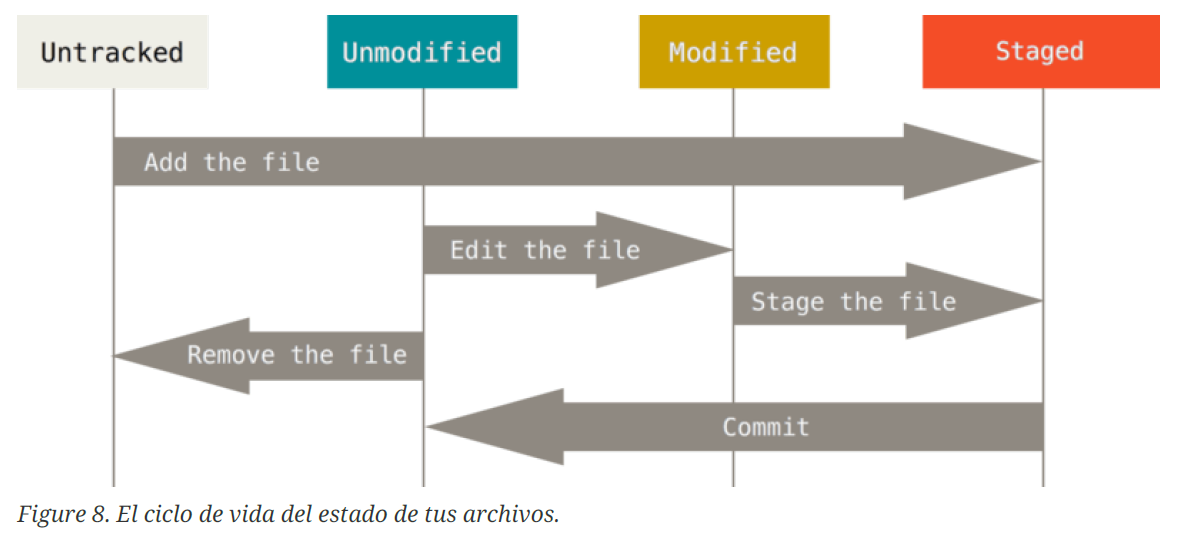
Este tiempo también depende de la capacidad de procesamiento y almacenamiento del hardware en el que se realizan las operaciones, y que tanta de esta capacidad se encuentra ocupada por otros procesos que realiza el mismo equipo.

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

La función implementada tiene un orden de complejidad de O(N). En este caso se organizan todas las películas en una lista y se recorre esta lista para encontrar los elementos que coincidan con el criterio de búsqueda. Dado que las películas están organizadas de acuerdo con la llave “**ID”** se tiene que recorrer toda la lista asegurando que no se omita ningún registro que coincida con el parámetro de búsqueda

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

Antes de hacer modificaciones al código se actualiza el repositorio local desde el repositorio central para que cualquier cambio que haya realizado otro miembro del equipo se vea reflejado en el ambiente local. De esta manera, posteriormente se realizan los cambios necesarios al código. Una vez realizados estos cambios se hace comital fork correspondiente. Se genera una pull request para agregar estos cambios al repositorio central. En caso de que no haya conflictos (y acorde al workflow acordado con el equipo), estos cambios se agregan al repositorio central. Lo anterior obliga a que los archivos tengan un clico de vida como se muestra a continuación.



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

La principal ventaja que tiene este tipo de trabajo es que facilita el trabajo colaborativo. Permite que todos los miembros de un equipo trabajen sobre la versión más reciente del código desde su propia Branch sin necesidad de comprometer el código del repositorio central. El control de tallado de cambios entre las versiones permite apreciar que se agregó/eliminó y facilita el rollback a versiones anteriores o la corrección de fragmentos particulares del código que puedan causar problemas.

**5. ¿Cuáles serían las buenas prácticas para solucionar conflictos?**

En este caso es necesario resolver el conflicto editando las secciones de código que están generando conflicto. Una vez se haya agregado/removido lo necesario es importante dejar un comentario sobre los cambios realizados en especial porque este tipo de modificaciones sobre los conflictos podrían generar problemas imprevistos sobre el Código y es importante poder hacer tracking apropiado de los problemas en caso de que se presenten.

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

Las funciones serían del orden O(N). se están leyendo elementos de una lista que escala linealmente con el número de elementos. El query solo tiene que leer los datos del CSV una vez para escribirlos y no es necesario que ejecute loops u operaciones que incrementarían el tiempo de ejecución

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

Se podría reducir la complejidad de la consulta si, por ejemplo, se solicitara el primer o el último elemento de la lista. En este caso la complejidad se convertiría al orden de O(1). Por otro lado, para aumentar la complejidad se podría solicitar consultas que reorganizaran el orden de la lista acorde con, por ejemplo, un criterio cronológico.

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

Un TAD define las propiedades particulares que tienen unas instancias, en este caso películas, sin definir explícitamente las operaciones que las afectan. Esto puede afectar la complejidad pues es necesario operar sobre este TAD con una serie de algoritmos que van a definir su complejidad.

La alternativa es un tipo de dato concreto CDT que incluye la implementación.