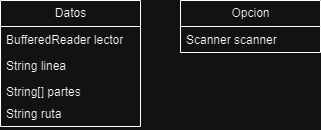
Diagrama UML



Para el funcionamiento de las funciones se utiliza una interfaz donde se inputa un numero para decidir que función se va a usar. a partir de acá se va a la clase funciones, que hace que se inpute la información necesaria para la consulta antes de efectuarla. Todas las consultas inician cargando los datos del archivo, separándolo en líneas y a esas líneas arreglándolas para la ejecución del programa, separándolas en un array. Se intento hacer este proceso una vez iniciado el programa y de esta forma reducir el tiempo de ejecución de todas las funciones, dado que no sería necesario repetir este paso, pero por problemas al implementarlos se decidió dejar de esta forma.

En los casos que se usan hash la razón de lo mismo es:

Por ser eficiente al buscar, O(1) esperado y O(n) en el peor caso

Por ser eficiente al insertar o actualizar valores, O(1) esperado y O(n) en el peor caso

fácil pasar del array a un heap en caso de necesitar ser ordenado, haciendo esto en O(nlog(n))

Función 1:

En esta funcion al cargar las distintas lineas, se guarda la informacion necesaria en un heap ordenado con raiz minima (nombre de la cancion, poscion). Ocurre un ordenamiento de las canciones al heap.

Tiempo de ejecucion y memoria alocada 1: 2429 ms | 4.51 GB 2: 2544 ms | 4,61 GB 3: 2312 ms | 4,61 GB

Función 2:

Se ejecuta la función, para cada líneas se compara la fecha y si lo esta se añade a un has que guarda el nombre de la canción y la cantidad de veces que ha aparecido, sumándole uno en caso de que ya esté en el hash. Después se guardan los datos en una heap, con la key siendo la cantidad de veces que aparecen las canciones. De esta forma ordenándose. Se imprimen las top 5 canciones

Tiempo de ejecución y memoria alocada 1: 2118 ms | 4.71 GB 2: 2436 ms | 4,61 GB 3: 2178 ms | 4,48 GB

Funcion 3:

En esta función se inicializa un hash que guarde el nombre de cada artista que aparece en el top como key, y la cantidad de apariciones. De la misma forma que en la función anterior se inicia en 1 al aparecer y se actualiza el valor del hash en cada pasada. En caso de que haya varios artistas se separa y se cuenta cada uno por separado, sumándole una aparición a todos. Cuando se termina el proceso, se guardan los elementos del hash en un heap para de esta forma poder ordenarlos según la key, la cual es cambiada y de esta forma es a cantidad de apariciones, ordenándolos por las mismas.

Tiempo de ejecución y memoria alocada (Todas las fechas) 1: 3422 ms | 5,84 GB 2: 3156 ms | 5,85 GB 3: 3001 ms | 5,95 GB

Funcion 4:

En esta función se inicia un contador en 0 se recorre la lista, revisando que la fecha y nombre de artista coincidan. En caso de lo que haga se incrementa el contador en 1.

Tiempo de ejecución y memoria alocada 1: 2174 ms | 4,61 GB 2: 2280 ms | 4,48 GB 3: 2257 ms | 4,68 GB

Función 5:

En esta función se inicializa un hash, se revisa que la fecha está en el rango buscado, en cual caso se revisa que el hash no contenga la canción, y el tempo este en el rango buscado. Si cumple todo esto se añade al hash. Se cuenta la cantidad de veces que se añadieron canciones al hash y se devuelve esto.

Tiempo de ejecución y memoria alocada 1: 2287 ms | 5,01 GB 2: 2820 ms | 5,01 GB 3: 2910 ms | 4,93 GB

Viendo los datos de memoria usada y tiempo demorado, se pudo notar que gran parte de este uso fue por la necesidad de cargar la lista en cada corrida de función, las distintas partes de las mismas tomaron la siguiente cantidad de tiempo en promedio. String Split 1,42 S | 1,84 GB Read Lines 339 ms | 322 MB Eliminar comillas 388 ms | 2,49 GB

Una posible implementación para esto podría haber sido crear un hash para las canciones, con un identificador (id) y una clase canción donde se guardaba la información que se podría necesitar. De esta forma no se necesitarían cargar tantos datos en cada ocasión y no se guardarían datos no usados.

Los datos de alocacion de memoria y tiempo de funcionamiento fueron recuperados con el IntelliJ Profiler. Especificaciones de la computadora usada. RAM usada: 16 gigas de ram DDR4 3200Mhz CPU usada: Intel i7-11370H, 330GHz