ANALISIS DE RESULTADOS

Sebastián Contreras Salazar Cod 202020903
Valentina Goyeneche Cod 201922380

Análisis en el rendimiento:

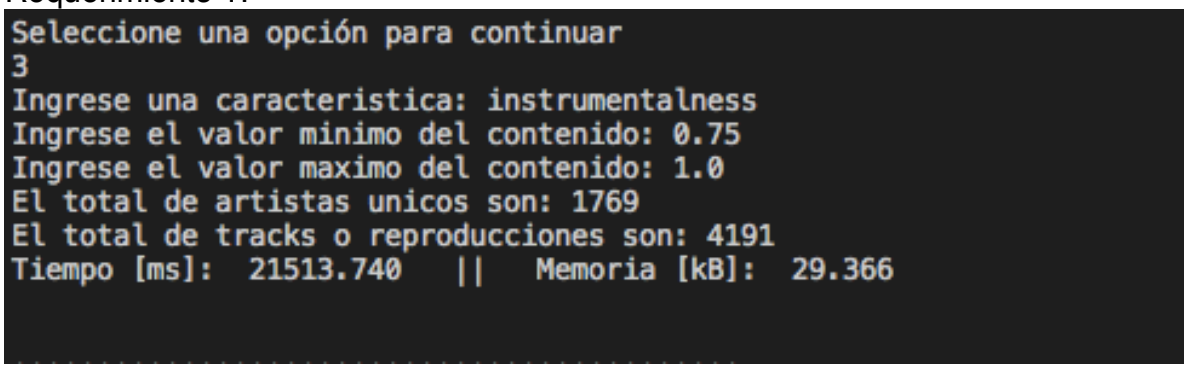
1) Carga del catálogo:

```
Seleccione una opción para continuar
2
Cargando información de los archivos ....
Hay 63398 tracks cargados.
Hay 10413 artistas unicos cargados.
Hay 63142 eventos cargados.
Tiempo [ms]:  2425529.798   ||   Memoria [kB]:  8447129.905
```

Para la carga del catalogo se uso la siguiente estructura: Un diccionario grande que va ser en si el catalogo, que cada llave dentro del diccionario van a ser las caracteristicas que se pueden encontrar en el csv del context, que son instrumentalness, speechiness, liveness, artist_id y created_at, entre otros, en el que cada una de estas caracteristicas son arboles organizados o ordered maps. Igualmente, junto con estos arboles se tiene que hay una lista de tracks en el que basicamente, son una lista de todos los tracks que se miran, con el fin de que en view se puedan mirar la cantidad de tracks cargados. Ahora bien, dentro de cada mapa organizado, se tienen como llaves las respectivas caracteristicas que se tienen dependiendo de la caracteristica del mapa organizado, en el que por ejemplo, en el mapa de instrumentalness se tienen adentro llaves que son 0.1,0.2, entre otros, que basicamente son los respectivos valores de los instrumentalness que se encuentran en los tracks, puestos en llaves. Ahora bien, como valores de esas llaves se tienen como una lista, mapas de tracks que cumplen con los respectivos valores dados por caracteristica y su llave, en el que siguiendo el ejemplo dicho, si se tiene instrumentalness de 0.1, se tendrian como valores de esas llaves una lista de mapas de los tracks que tinene instrumentalness de 0.1 y así. Igulamente, cabe resaltar que para esta carga se neecsitaban pasar los datos a int o floats, puesto que es ideal para comparar datos y no tener que comparar datos con un orden eterolexico (de strs) que realmente no es igual que con los enteros y los floats. Teniendo eso en cuenta, al añadir los datos, se puede observar que son un poco lentos, por lo que se tiene un tiempo que supera los 2 millones de ms y la memoria tambien es bastante grande y todo eso se deriva por la cantidad de arboles que se tienen. Ahora bien, es necesario resaltar que se tuvo la idea de que se podrian hacer más recorridos aqui, en el que una idea inicial era crear listas en donde guardabamos todos los hashtags de sentiment_value completo y otra lista para guardar los datos del user_track_hashtag, con el fin de añadirlos a cada track respectivo y añadirle su vader_avg y su respectivo hashtag, pero se descarto esta idea, porque nuestro catalogo se demoraria más porque sería más complejo, sin embargo, se sabe que se debian haber añadido aquí, puesto que

si se usan datos más grandes se puede ver que los requerimientos no son lo mismo. En cuanto a la complejidad, se tiene una complejidad de o(n), puesto que se miran cada track. Finalmente, en cuanto a lo que imprime aquí, se puede observar que con el archivo small se cargaron la cantidad de tracks correctamente, se cargaron los artistas unicos y los eventos, siguiendo la idea de que created_at son los eventos y los artistas unicos se pueden sacar con la función size del organized map del artist_id.

2) Requerimiento 1:



Para el requerimiento 1, usando el catalogo de la carga del catalogo, como se puede ver tiene un tiempo en relación a la carga, considerablemente menor al igual que la memoria, puesto que en estos casos se tiene una complejidad de o(log(n)), o(nlog(n)) o o(#elementos que estan en el rango) en el mejor caso y como un caso promedio, y se tiene o(n^2) en el peor caso. Esto se debe a que al usar la función de om.values() permite que se use solo una cantidad especifica de datos, por lo que no se tienen que ver todos los datos o tracks, causando así que se limite mucho la cantidad de datos a iterar. Igualmente, en este caso al usar dos while, posiblemente se pueda ver como algo muy demorado, pero no es correcto, porque es con los dos while que se miran todos los tracks dentro del rango en el que en el primer while se saca cada valor de la caracteristica de los om (0.1,0.2, entre otros de la caracteristica vista (instrumentalness, tempo, entre otros)), osea las caracerteristicas respectivas, y dentro del segundo while se sacan cada mapa de la lista y se sacan sus valores, en el que en este caso para este requerimiento se usan unas listas para poder mirar si los artist_id estan ahí, con el fn de poder sacar el total de artistas uncos, y para las reproducciones se sacan a partir del tamaño de la lista sacada o de los valores o caracteristicas dentro del rango seleccionado en forma de lista. Al final se devuelve la suma de los tamaños de cada lista del primer while y el tamaño de la lista de los artist_id. Al final se tendria que hay un total de artistas unicos de 1769, que es el esperado, pero el total de tracks o reproducciones no son exactos, posiblemente por no usar los 3 csvs en la carga inicial.

3) Requerimiento 2:

```
4
Ingrese el valor minimo de la caracteristica Energy: 0.5
Ingrese el valor maximo de la caracteristica Energy: 0.75
Ingrese el valor minimo de la caracteristica Danceability: 0.75
Ingrese el valor maximo de la caracteristica Danceability: 1.0
Energy esta entre 0.5 y 0.75
Danceability esta entre 0.75 y 1.0
Los track unicos en eventos son: 1703
**********************************************
Track 1:47389c49585fe5d8cc97649243722613 with energy of 0.704 and danceability of 0.758
**********************************************
Track 2:50629e7717bf88fc57f52b4dc244d44a with energy of 0.563 and danceability of 0.765
**********************************************
Track 3:67161e17a4cfcfe141903fcdbbc67b42 with energy of 0.636 and danceability of 0.794
**********************************************
Track 4:744e034093646721870b76b70d2b65e4 with energy of 0.51 and danceability of 0.859
**********************************************
Track 5:c88c4697c3c73cd5c87eddcb7c98bd67 with energy of 0.577 and danceability of 0.826
Tiempo [ms]:  57542.220   ||   Memoria [kB]:  400.569
```

Para este requerimiento se usaron los mismos dos while que al anterior requerimiento, pero las tres grandes diferencias de este con el requerimiento 1 es que se usaron los om.values() para poder sacar una cantidad de valores que esta dentro del rango de las caracteristicas de energy y danceabilty, en el que no se usan en este caso las cantidades sumadas y los artistas unicos, sino que se usa otro if para cuando se llega a cada mapa de cada track, para que en el primer while, donde se esta iterando la lista del om.values() de energy, se sacan los datos de danceability y de track_id con el fin de hacer dos ifs, un primer if para poder mirar si esta dentro del rango del danceability ese track sacado y el otro if para mirar si ese track_id ya se vio, para no repetir reproducciones o tracks. Ahora bien, una vez se iteran esos dos while o mejor dicho se itera la lista del om.values() de energy, se hace lo mismo, pero con la lista de danceability, sin embargo la lista usada para mirar si el track_id esta presente o ya se vio y la lista para guardar los tracks en mapas(que se guardan si se cumple la condición de que estan dentro de los rangos de energy y danceability y no se ha visto su track) se usan de nuevo. Dentro de este while final, se hace lo mismo que el ultimo while que se uso para iterar la lsita de los energy. Ahora bien, en este caso en vez de mirar y sacar el danceability se sacan los energy, para mirar que los tracks sacados del rango de danceability estan tambien edntro del rango de energy, y si lo estan se vuelve a mirar si ya se vio ese track, y sino se ha visto, se añade a la lista de tracks. Todo esto tiene complejidad tambien de o(nlog(n)) como en un caso promedio porque esos om.values(), de nuevo, nos permiten trabajar con solo una cantidad de datos. En cuanto a los resultados obtenidos, se puede observar que se obtuvieron los tracks unicos exactos dentro del rango de ejemplo en el enunciado. Por otro lado, el tiempo y memoria son pequeños, posiblemente por la cantidad de datos que se devuelven, usan y trabajan. Finalmente, apra imprimir los tracks, se usa la función random() para poder imprimir tracks aleatorios.

4) Requerimiento 3:

```
Seleccione una opción para continuar
5
Ingrese el valor minimo de la caracteristica instrumentalness: 0.6
Ingrese el valor maximo de la caracteristica instrumentalness: 0.9
Ingrese el valor minimo de la caracteristica tempo: 40.0
Ingrese el valor maximo de la caracteristica tempo: 60.0
Instrumentalness esta entre 0.6 y 0.9
Tempo esta entre 40.0 y 60.0
Los track unicos en eventos son: 9
**********************************************
Track 1:1e32c4e7e61870f300f9362f42e7751b with instrumentalness of 0.754 and tempo of 59.35
**********************************************
Track 2:4978ab7dad5dc1d843af6b3b422a8692 with instrumentalness of 0.755 and tempo of 56.228
**********************************************
Track 3:1e32c4e7e61870f300f9362f42e7751b with instrumentalness of 0.754 and tempo of 59.35
**********************************************
Track 4:d5470e12b055aeb25ec11efcc474f8bd with instrumentalness of 0.893 and tempo of 53.203
**********************************************
Track 5:7d323c9286f68ac35b86392b8a329bc2 with instrumentalness of 0.76 and tempo of 58.993
Tiempo [ms]:  10285.254   ||   Memoria [kB]:  11.947
```

Para este requerimiento se hizo basicamente lo mismo que el requerimiento 2, con la diferencia de que se usaron en este caso tempo e instrumentalness, en el que para los primeros dos whiles se uso la lista del om.values() del tempo y para los dos ultimos whiles se uso la lista del om.values de instrumentalness. En este requerimiento se siguieron, basicamente, los mismos pasos que en el requerimiento 2, puesto que aquí tambien se crean dos listas, se usan dos ifs para mirar que no se repitan tracks y que los tracks esten dentro del rango dado por el usuario. La complejidad es de o(nlog(n)) en un caso promedio, puesto que al igual que el req 2, al usar om.values() se miran solo una cantidad de datos especifica y no todos los datos completos. Ahora bien, en relación a los resultados, se obtuvieron los mismos tracks unicos que en el enunciado. Por otro lado, el tiempo y memoria son pequeños, posiblemente por la cantidad de datos que se devuelven, usan y trabajan. Finalmente, apra imprimir los tracks, se usa la función random() para poder imprimir tracks aleatorios.

5) Requerimiento 4:

```
6
Para buscar algun genero conocido (F) o desea crear un nuevo genero (V): V
Ingrese el nuevo nombre del genero: Violet Evergarden
Ingrese el valor minimo del tempo: 10.0
Ingrese el valor maximo del tempo: 40.0
**********************************************
**********************************************
Para el nuevo genero: Violet Evergarden se tiene que hay 5 artistas y 7 reproducciones
Donde el tempo minimo es 10.0 y el maximo es 40.0
Arist 1:ac0dd5665472427d78009b1c8a6ffca3
Arist 2:5d10604810aa41e3dfc37341038080f0
Arist 3:9abaead6f3119635e4fe4f2cbf1d7a5b
Arist 4:5d10604810aa41e3dfc37341038080f0
Arist 5:9abaead6f3119635e4fe4f2cbf1d7a5b
Arist 6:9abaead6f3119635e4fe4f2cbf1d7a5b
Arist 7:5d10604810aa41e3dfc37341038080f0
Arist 8:ac0dd5665472427d78009b1c8a6ffca3
Arist 9:9abaead6f3119635e4fe4f2cbf1d7a5b
Arist 10:a739d0298d7abb09f2e16c2d21e59bae
Tiempo [ms]:  68.246   ||   Memoria [kB]:  1.008
```

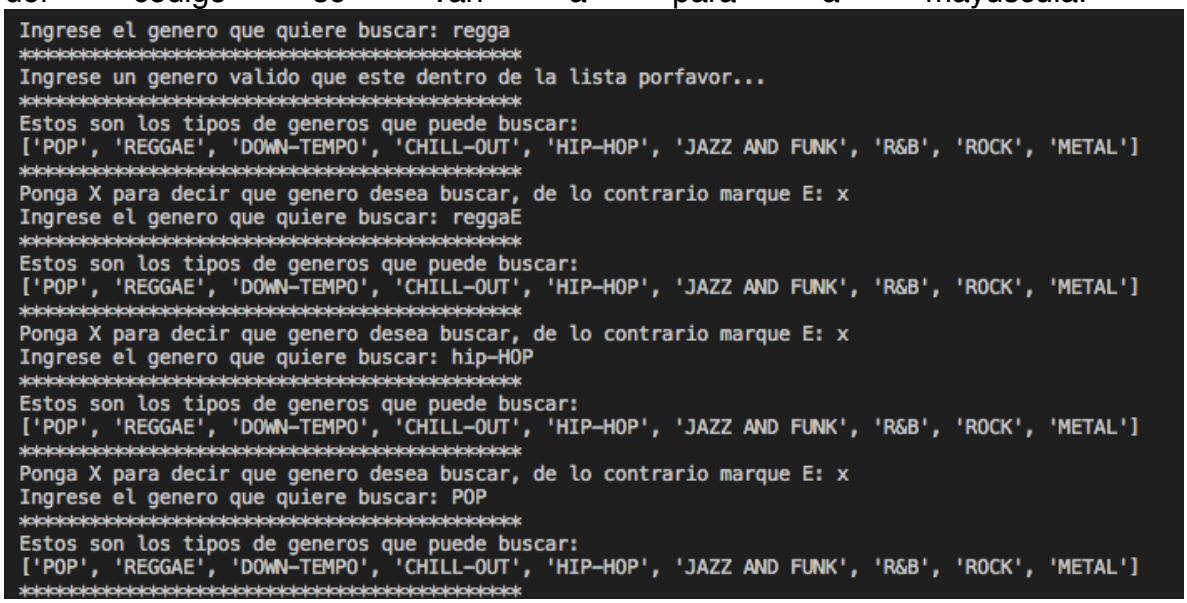
Para este requerimiento se hicieron muchos cambios y se hizo una interacción con el usuario única, en que inicialmente al seleccionar este requerimiento se

preguntan que marque lo que desea hacer el usuario, en donde si selecciona F se van a buscar y mirar géneros existentes, pero si selecciona V, u otra letra, se puede crear un nuevo género en el que el usuario elige el tempo del mismo. Como se puede ver en este caso, al seleccionar V, se le pide al usuario el nombre del género que desea crear y se le pide el valor mínimo y máximo de tempo para poder sacar los tracks que tienen el tempo dentro de ese rango. Ahora bien, en este caso se puede observar que se nombre el género “Violet Evergarden” y se puso un tempo mínimo de 10 y de 40, en el que en este caso se usa la función del requerimiento 4 que es exactamente la misma que la del req 1 con la excepción que devuelve la cantidad de artistas únicos, la lista de maps de tracks y la cantidad de tracks que hay, en el que en este caso al usar om.values() su complejidad de nuevo va ser o(nlog(n)), porque se trabajan con solo una cantidad de tracks específicos y no todos. Teniendo eso en cuenta, después de pasar por esa función de req 4, se devuelven tracks aleatorios, la cantidad de artistas únicos y reproducciones en total, en el que en este caso “Violet Evergarden” tiene 5 artistas únicos y 7 reproducciones. Finalmente, se puede observar que el tiempo es bastante pequeño al igual que la memoria, posiblemente, de nuevo, por la cantidad de datos que se encontraron dentro de ese rango del tempo.

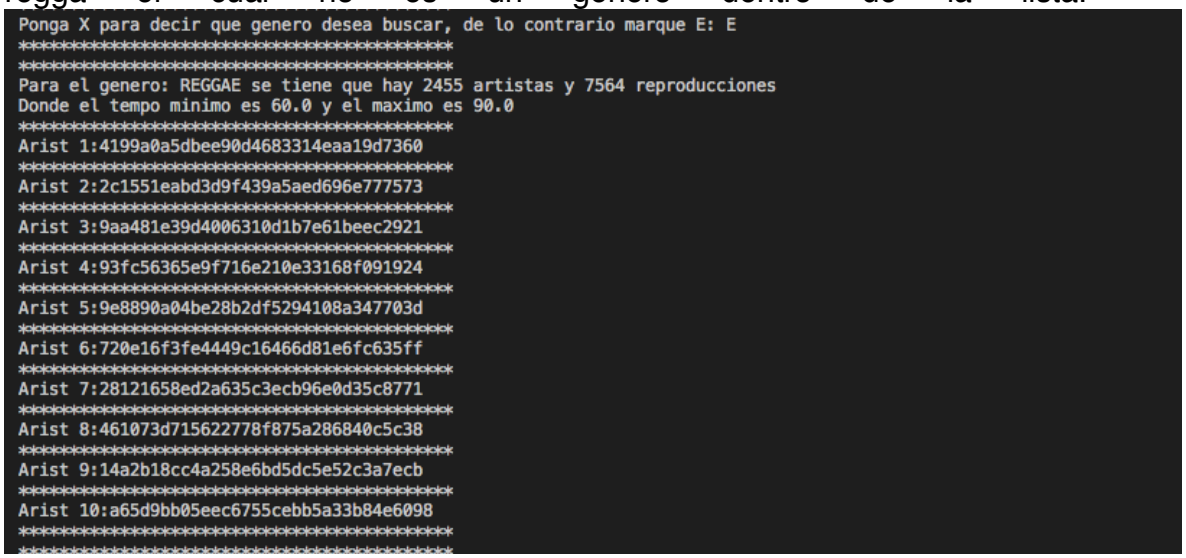
```
**********
Seleccione una opción para continuar
6
Para buscar algun genero conocido (F) o desea crear un nuevo genero (V): F
**********
Recuerde poner los generos que se desea buscar en mayuscula...
**********
Estos son los tipos de generos que puede buscar:
['POP', 'REGGAE', 'DOWN-TEMPO', 'CHILL-OUT', 'HIP-HOP', 'JAZZ AND FUNK', 'R&B', 'ROCK', 'METAL']
**********
Ponga X para decir que genero desea buscar, de lo contrario marque E: x
Ingrese el genero que quiere buscar: regga
**********
Ingrese un genero valido que este dentro de la lista porfavor...
**********
Estos son los tipos de generos que puede buscar:
['POP', 'REGGAE', 'DOWN-TEMPO', 'CHILL-OUT', 'HIP-HOP', 'JAZZ AND FUNK', 'R&B', 'ROCK', 'METAL']
**********
Ponga X para decir que genero desea buscar, de lo contrario marque E: x
Ingrese el genero que quiere buscar: reggaE
**********
Estos son los tipos de generos que puede buscar:
['POP', 'REGGAE', 'DOWN-TEMPO', 'CHILL-OUT', 'HIP-HOP', 'JAZZ AND FUNK', 'R&B', 'ROCK', 'METAL']
**********
Ponga X para decir que genero desea buscar, de lo contrario marque E: x
```

Ahora bien, si se selecciona la opción F, se puede observar que se van a imprimir una lista de los géneros que puede desear buscar el usuario, en el que puede buscar ya sea 1 o todos, en donde para ello tiene que seleccionar uno a la vez, puesto que se va creando una lista de los géneros que desea buscar el usuario. Si se selecciona la opción X en minúscula o mayúscula, se va a pedir el nombre del género que desea buscar al usuario, pero si elige la opción E, se va terminar de buscar y se van a imprimir los datos de información de los 3 géneros. Ahora bien, realmente no es necesario escribir los nombres de los géneros en mayúscula o minúscula, puesto que dentro

del código se van a para a mayúscula.



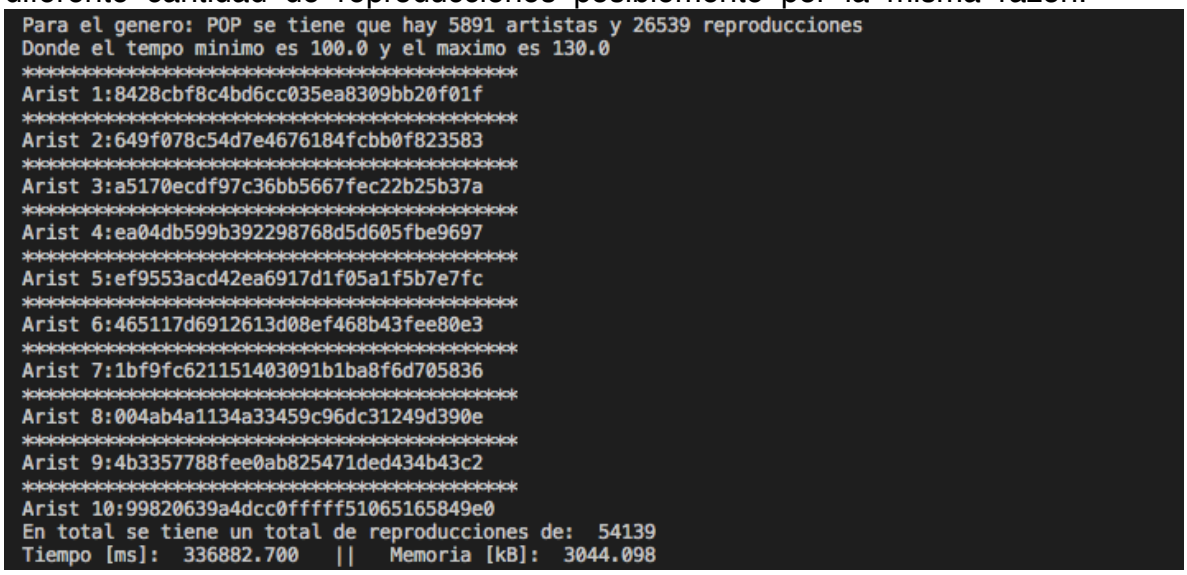
Sin embargo, si el usuario escribe mal el nombre del género o no esta ese género dentro de la lista que se le proporciona al usuario de los géneros que puede buscar, se le va a imprimir un mensaje de que debe ingresar un género valido que este dentro de la lista, por lo que vuelve a iterar hasta que el usuario ingresa E u otra letra, como se puede ver con el caso de que escribió regga el cual no es un género dentro de la lista.



Una vez, se haga la iteración completa, se van a empezar a mirar por cada género, en el que en este caso se seleccionaron reggae, hip-hop y pop en ese orden. Ahora bien, es necesario resaltar que se imprime 1 genero a la vez, pero el tiempo y la memoria se van sumando, puesto que se usa el mismo requerimiento 4 que se usó para crear un nuevo género, pero esta vez se crean unas nuevas variables para sumarles el tiempo y la memoria y devolverlo completo al final. Del mismo modo, en este caso de reggae, siguiendo con el resultado del enunciado, se puede observar que se

obtuvieron la misma cantidad de artistas, pero una diferente cantidad de reproducciones, puesto que posiblemente al no usar los 3 csvs el resultado de reproducciones cambia.

Este es el resultado obtenido de hip-hop, que también se puede observar que en relación al enunciado, se tienen la misma cantidad de artistas, pero diferente cantidad de reproducciones posiblemente por la misma razón.

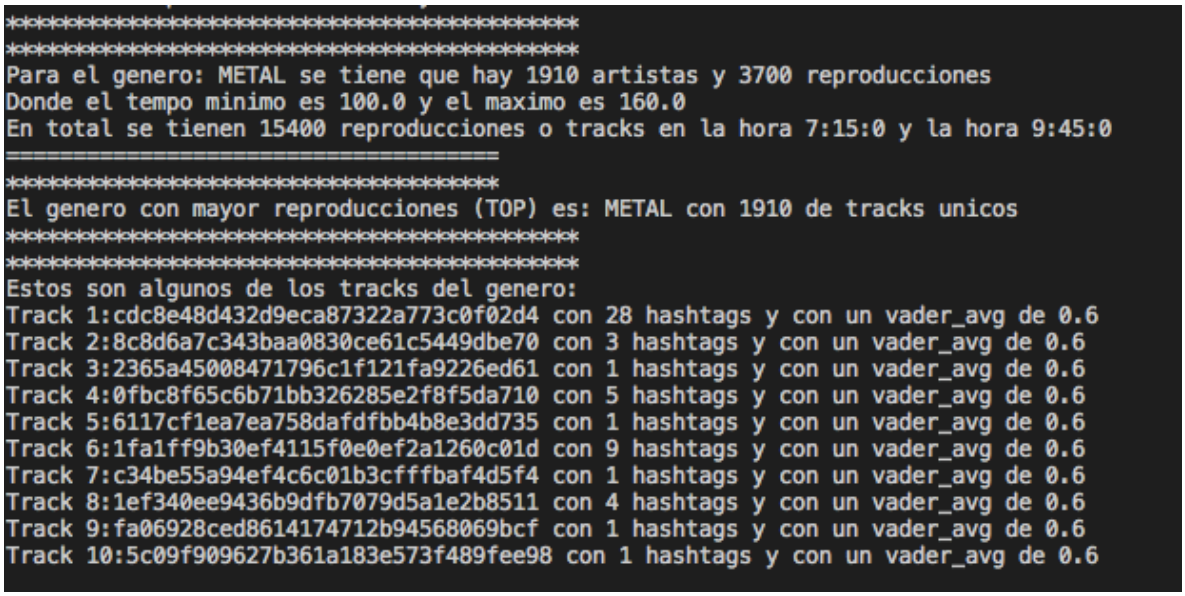


Este es el resultado de pop, que, de nuevo, tiene la misma cantidad de artistas, pero las reproducciones son diferentes y son mayores, por la misma razón posiblemente. Ahora bien, se puede observar que al final se muestra el total de reproducciones, que evidentemente no es el mismo, pero realmente no está demasiado lejano del que se proporcionó en el enunciado. Para poder solucionar estos problemas, posiblemente, se pueda hacerlo si en el catálogo se añaden únicamente los tracks que tienen un vader_avg, que tengan valores esos vader_avg y que tengan hashtags. Finalmente,

creemos que la complejidad de esto es también o(nlog(n)x # de géneros a observar o buscar), puesto que entre más géneros se desean buscar, más memoria y tiempo se demora. En cuanto al tiempo y memoria, se puede observar, que realmente, la memoria no es muy grande en relación con el catalogo, puesto que se usan solo unos datos específicos y no todos los tracks y datos de los árboles de tempo, en este caso específicamente. Sin embargo, el tiempo si es considerablemente grande en relación con los demás requerimientos vistos, puesto que se miran más datos y se hacen más recorridos, por lo que el tiempo es más largo.

6) Requerimiento 5:

Para este requerimiento 5 se tiene una complejidad de o(n+#elementos del rango de horas(nlog(n))), puesto que se mira por completo toda la lista de los hashtags para poder buscar todos los hashtags que tienen el mismo track_id, para poder encontrar la cantidad de hashtags y luego poder encontrar el promedio de los vader_avg. Ahora bien, también se hace un recorrido, que depende de la posición del hashtag, en sentiment_values, por lo que el recorrido que se pueda hacer a la lista de sentiment puede ser largo o pequeño. Por otro lado, se podría decir que ningún resultado de reproducciones es el correcto, posiblemente, porque no se juntaron los datos entre los tres csvs desde el inicio. Ahora bien, para este requerimiento se ahorró código al reusar el código del requerimiento 4.



Sin embargo, se obtiene correctamente al final que Metal, a pesar de que no tienen las reproducciones exactas al del enunciado, se obtiene que es el género con mayores reproducciones. Del mismo modo, al devolver los tracks podemos devolver la cantidad de hashtags y el vader promedio correctamente, puesto que cuando usamos la función de catalogo2, se crea un nuevo árbol organizado de la cantidad de hashtags que hay y también, se crea un nuevo árbol organizado para los vader_avg, con el fin de poder devolverlos. Ahora bien, se sacan la cantidad al mirar cada track y contar cuántos de esos tracks tienen un vader_avg, con el fin de contar esos hashtags por track_id y poder sumar todos los vader_avg, para poder sacar el promedio final y devolverlo correctamente.