Departamento de Ingeniería de Sistemas y Computación
Estructuras de Datos Y algoritmos
ISIS1225 - 202119

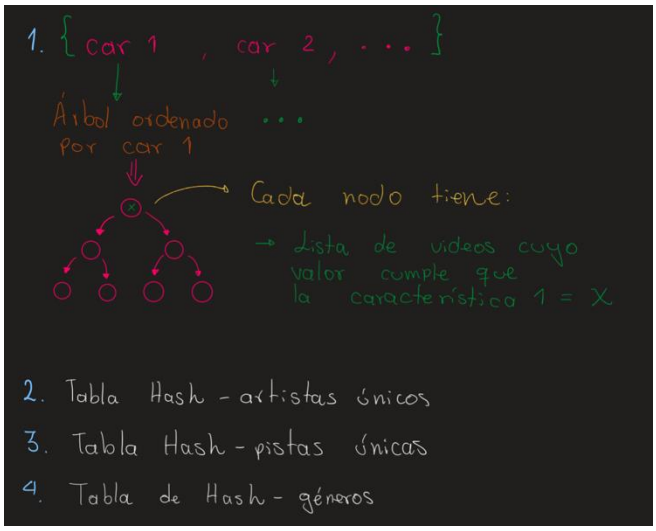
Documento de Análisis Reto 3

Ana Sofía Castellanos Mosquera
202114167
a.castellanosm@uniandes.edu.co

Martín Santiago Galván Castro
201911013
ms.galvan@uniandes.edu.co

Análisis General – Guardar datos:

Para el cumplimiento de los requerimientos y el uso de las estructuras de datos. Primero, se creó un catálogo en donde se almacena toda la información a usar. Este catálogo contiene 5 elementos. El primero es un diccionario en donde se guardan índices de características de contenido por el valor. Se hubiera podido usar una tabla de hash para este elemento, pero como ya se conocían las llaves que se iban a usar, se optó por un diccionario. El segundo elemento es una lista que se implementa como arreglo, aquí se almacenan todas las canciones que se leen. El tercer y cuarto elemento son tablas de hash implementadas con mecanismo de colisiones probing. Estas tablas de hash guardan cada registro de canción y autor único. Sus llaves son ya sea el id de canción o el id del artista. Por último, se tiene un mapa como tabla de hash tipo probing el cual guarda los géneros junto con su asignación de bpm necesario para cumplir el requerimiento 4.



Para poder ahorrar parte de memoria al ejecutar el programa, solo se guardan ciertas características del archivo de Excel que se lee. Estos son el id de canción, el id de artista y las características de contenido relevantes para resolver los requerimientos.

Siendo más específicos en cómo funciona el índice por características de contenido. Dentro de cada elemento del diccionario por características, se encuentra un árbol rojo negro. Cada nodo tiene como elemento de clasificación el valor de la característica en la cual se encuentra en el diccionario del. Cada vez que se guarda un nuevo elemento que se lee, se actualizan y añaden en todos los árboles de este diccionario de conformidad al valor que tienen en la característica de un árbol. En cada árbol con el propósito de reducir el número de nodos, no se guardan con todas las cifras significativas de la característica por la que se clasifica el árbol sino por intervalos, con una sensibilidad de 2 decimales lo cual reduce la cantidad de nodos a 100 en aquellos clasificados por características con intervalos entre 0 y 1. (Ver análisis de requerimiento 1 para más detalles).

Por último, sobre la tabla de hash del requerimiento 4, este se crea con las llaves siendo el nombre de un género. Y el valor es una tupla de dos elementos. Estos elementos son los rangos de BPM del género. Esta estructura se guardó así, ya que como parte del requerimiento 4, el usuario debe de ser capaz de ingresar nuevos géneros a la lista de géneros.

Análisis de complejidades:

Requerimiento 1 (Equipo de trabajo):

El requerimiento 1 solicita la búsqueda de canciones a partir de la intersección de características que son seleccionadas por el usuario. Para ello considerando que los datos se encuentran cargados dentro de un diccionario que tiene como llaves las características y como valores árboles binarios, el requerimiento N1 cuenta con el siguiente algoritmo:

1. Ingresar al diccionario de características de contenido
2. Obtener de ese diccionario el árbol binario que está organizado por la primera característica que se ingresa por parámetro
3. Obtener una lista con todos los valores que se encuentran dentro de los nodos del árbol que están en el rango ingresado por parámetro de dicha característica, esto es una lista que contiene otras listas con las canciones clasificadas por su valor en la característica 1
4. Realizar un recorrido por la lista de listas (lista de nodos) y por la lista de elementos que está presente en cada una de esas listas.
5. En dicho recorrido se verifica si el elemento que ya cumple con el rango para la característica N1 cumple con estar en el rango dado para la característica 2. De ser así se aumenta el contador
6. Debido a que el requerimiento también solicita la cantidad de artistas únicos, se realiza una tabla de hash que añade los artistas como llaves de forma única y se verifica el autor para cada reproducción y se revisa si dicho elemento ya está en la tabla o no para añadirlo.

Para este requerimiento obtener el árbol ordenado a partir de la primera característica se realiza en O(1).

Debido a que el árbol está guardado por rangos de valores y no por nodos con la cantidad precisa de cifras significativas de cada reproducción, permite conocer que en el caso de las características que cuentan con valores entre 0 y 1 en el peor caso existirán 100 nodos (con una sensibilidad del rango de dos decimales) que resulta de ver la cantidad de decimales con dos cifras que estén entre 0 y uno. Considerando esto, en el peor caso del algoritmo (elección de los rangos entre 0 y 1) se obtendrá una lista de listas con la función values(…) con 100 elementos.

De esta manera, el primer ciclo que ingresa a los elementos de la lista de listas ingresará para las características entre 0 y 1 un máximo de 100 veces en el peor caso. Luego, el segundo ciclo debido a que se pueden revisar todos los elementos del archivo cuenta con una complejidad de O(N). Dentro del segundo ciclo, las operaciones realizadas cuentan todas con una complejidad de O(1), ya que se verifica si el video de dicha lista cumple con que su característica N2 este entre el rango dado por parámetro y se revisa si el artista ya está en el Map con la operación contains(...) la cual es O(1) y si no está en el Map se añade con la operación put(…) que es también O(1). Finalmente se revisa el tamaño del Map de artistas con size(…) lo cual es O(1).

A partir de lo anterior, generalizando la cantidad de nodos que puede llegar a contener el árbol cuyas características están entre 0 y 1 a partir de la sensibilidad se tiene que es:

$$Nodes = 10^{cantidad\ de\ decimales}$$

En el caso de la carga de datos implementada la cantidad de decimales es de 2 con lo cual la cantidad de nodos es 100.

Esta cantidad de nodos nos permite saber la ecuación de complejidad para el doble ciclo el cual es de:

$$O(Nodes) * [O(N) + Complejidad\ operaciones\ ciclo\ 2]$$

La cantidad de nodos en las características entre 0 y 1 es de 100 con lo cual la complejidad del ciclo uno se reduce a O(1) es decir constante y queda la complejidad total como:

$$O(N) + Operciones\ ciclo\ 2$$

Como se analizó anteriormente las operaciones que se encuentran dentro del segundo ciclo son todas de O(1) con lo cual la se tiene que la ecuación Big O del requerimiento 1 es de:

$$O(N)$$

Requerimiento 2 (Martín Santiago Galván Castro):

En el caso del requerimiento 2, primero, al igual que en el requerimiento 1, se hace búsqueda por dos características y los rangos de estas. Pero a diferencia del requerimiento 1, se busca en dos categorías específicas y solo se le pregunta los límites de esta. Adicionalmente, no se busca el número de eventos de reproducción si no que se buscan pistas que estén en dichos rangos.

Se ingresa a los mapas ordenados de manera similar a la del requerimiento 1. Primero se hace un values, para obtener la lista de nodos organizados para la primera característica. Luego se revisa las listas dentro de los nodos buscando eventos de canción que estén dentro del rango de la característica dos. Esto se hace con un if. Posteriormente, se revisa si el id de dicho evento se encuentra como llave en una table de hash. Si lo está, no se hace nada, y si no está, se agrega a la tabla de hash usando como llave el id de canción y como valor un diccionario que guarda solamente el id de canción, y los valores de las dos características.

Las dos características son liveness y speechness. Se ingresa al árbol de speechness primero, y luego se revisa con el if la característica liveness.

Después de haber realizado la anterior iteración, se guarda el tamaño de la tabla de hash y se hace una lista de los valores de la misma tabla usando setValues(). Después, utilizando randint(…) para generar un numero entre 1 y el tamaño de la anterior lista. Se busca seleccionar con esto 8 posiciones de la lista de setValues() que no se repitan para guardarlos en una lista y devolverla.

Al final, se devolvería el tamaño de la tabla de hash, ósea el número de canciones únicas encontradas y una lista que contiene 8 de esas canciones elegidas de manera aleatoria. Más específicamente, se guarda el id de la canción, y los valores de la característica.

Para el análisis de complejidad, se puede escribir de la siguiente manera:

$$O(1) \times \big(O(n) \times O(1)\big) + O(n)$$

La explicación de la anterior expresión empieza desde el primer O(1) que funciona así porque hay una máxima cantidad de nodos en cada árbol dado a que se guardan por intervalos. La multiplicación posterior es por la iteración en cada nodo, recorriendo los elementos de la lista dentro de un nodo. El O(1) posterior multiplicando el anterior se debe al contains del hash y luego a agregar un elemento a la tabla de hash. Luego, la suma de O(n) se debe a la realización del setValues y luego el seleccionar las 8 posiciones aleatorias. Al final, por la notación Big O se termina describiendo la complejidad del requerimiento como:

$$O(n)$$

Requerimiento 3 (Ana Sofia Castellanos Mosquera):

Para el requerimiento 3, se necesitaba realizar la intersección de las características Valencia y Tempo, para ello se siguió la siguiente estrategia:

1. Ingresar al diccionario de los árboles binarios por característica
2. Crear un Map de canciones únicas y una lista que contendrá la información requerida de 8 canciones que cumplen con los rangos de ambas características
3. Acceder al mapa binario por la característica valencia y obtener los nodos que cumplen con el rango de valencia
4. Realizar un recorrido por la lista de listas (lista de nodos) y por la lista de elementos que está presente en cada una de esas listas.

5. En dicho recorrido se verifica si el elemento que ya cumple con el rango para la valencia cumple con estar en el rango del Tempo. De ser así se verifica si la reproducción ya esta en el Map de canciones únicas, de añade si no esta y se añade a la lista de retorno si el tamaño de la lista no ha llegado a 8.

Dentro del algoritmo que se siguió para resolver el requerimiento cabe mencionar que primero se accede al árbol binario por la valencia puesto que esta se conoce de antemano cuenta con valores entre 0 y 1 y como se vió en el primer requerimiento se tendrán como máximo 100 nodos. Caso contrario al del Tempo donde existirán más nodos puesto que tiene valores que pueden ser menores a 20 bpm y mayores a 200 bpm, y no solo enteros sino también decimales de dos cifras.

Partiendo de acceder primero a la valencia se obtendrá una lista de listas en el peor caso de 100 elementos a partir de la operación values(…) que equivale a la cantidad de veces que se ingresará al ciclo 1. Luego, se ingresa a los elementos de cada lista dentro de la lista de listas lo cual tiene una complejidad máxima de O(N). Así mismo las operaciones dentro del ciclo corresponden a contains(…) para verificar si la canción está en el Map, put(..) para añadirla en caso de que no este y las operaciones size(…) para ver que la lista no excede los 8 elementos y addLast(…) para añadir dicha canción a la lista. Estas operaciones tienen todas unas complejidades de O(1).

Considerando lo anterior, la compleidad del requerimiento 3 en total es la complejidad de values que es constante puesto que la cantidad de nodos para la valencia es de máximo 10, sumada con la complejidad del doble ciclo que es O(N).

La ecuación que representa lo anterior es:

$$O(100) + O(100) * (O(N) * \left[O(1) + O(1) + O(1) + O(1) + O(1)\right))$$

$$O(100) + [O(100) * O(N)]$$

Debido a que 100 es en realidad una constante se tiene que la ecuación es:

$$O(1) + (O(1) * O(N)$$

Como se considera solo el de mayor complejidad, la complejidad total es de:

$$O(N)$$

Requerimiento 4 (Equipo de trabajo):

Para lograr el requerimiento 4, el cual necesitaba de varias cosas y implementaciones además de solo revisar la estructura de datos. Se realizó buscando resolver lo siguiente:

- Se recibe como entrada una lista de géneros para realizar la búsqueda
- Los géneros que se pueden usar como opción ya están definidos, y cada genero tiene un rango de BPM por el cual busca

- El requerimiento debe de permitir ingresar un nuevo género a los géneros que se pueden buscar. Para esto se pide el nombre único para el nuevo género musical, el valor mínimo del tempo y el valor máximo del tempo
- Se debe de devolver el total de eventos de escucha que se encontraron, el total de eventos de escucha por género, el total de artistas únicos por género y una lista de los 10 primeros artistas por género.

Para resolver el primero, se realizó una lista como arreglo en el view, en el cual el usuario puede agregar género que desea agregar en la búsqueda. Para el segundo, se creó en el catálogo una tabla de hash en donde las llaves de las parejas eran los nombres de los géneros y el valor una tupla que contiene el rango. Se decidió usar una tabla de hash en vez de un diccionario para esta estructura dado a que era necesario preguntar al usuario por entradas nuevas. Estos problemas, sin embargo, no afectan en cómo se desarrolló el algoritmo para realizar una búsqueda.

El algoritmo para iniciar la búsqueda de un género, parte después de encontrar los rangos de tempo para dicho género. Sabiendo esto, se aplica la operación values(…) de la librería de mapas ordenados, que tiene como entrada el árbol sobre el que se busca, un límite inferior de llave y un límite superior de llave. Esta operación tiene como complejidad big O de O(N), en donde N en este caso es el número de nodos del árbol. Lo anterior se debe a que, en el peor de los casos, todos los elementos del árbol se encuentran en el rango. Sin embargo, dado a que se guardan por intervalos los elementos en el árbol, se puede asegurar que incluso en el peor de los casos, el número de nodos va a ser constante. Por lo que la complejidad de esta operación será constante o O(1). Al realizar esto se devuelve una lista de valores del árbol. Dentro de estos valores, se encuentran listas de eventos de reproducción. Se recorre estas listas contando el número de eventos de reproducción que se encuentran, adicionalmente, se revisa por medio de una tabla de hash que artistas son únicos. Esto se logra usando como llaves de una tabla de hash el id de artista. Se guardó como valor un 1 en estas parejas pues el valor no importa. Al finalizar la iteración, solo se devuelve el tamaño de la tabla de hash Para tomar los primeros 10 artistas que se encuentran, cuando se ve que un id de artista no está en la tabla de hash, se agrega ese id y se guardan los 10 primeros registros que se encuentran.

Ahora, habiendo explicado y realizado el algoritmo para encontrar la información de un género, se itera sobre una lista con nombres de género realizando dicho algoritmo. Se escribió el código de forma que lo que se devuelve después de esta iteración sea un diccionario con un elemento cuyo valor es el total de eventos de reproducción que se encontraron. Y otro cuyo valor es una lista que tiene los resultados de búsqueda para cada uno de los géneros.

Teniendo lo anterior, se imprime la información de manera organizada en el view.

Para el análisis de complejidad, se tiene la suma de las complejidades de cada operación del requerimiento. Aquí, se toma en cuenta lo necesario para la búsqueda y no la parte de obtener los limites por género o agregar un género:

$$O(1) \times \Big(O(n) \times \big(O(1)\big)\Big)$$

La anterior expresión de complejidad se escribe de dicha manera por la iteración doble que sucede por iterar una lista que contiene listas. El primer O(1) es por values, que tiene una complejidad constante por las razones que se mencionaron anteriormente. El O(n) es por recorrer una lista que contiene los eventos de reproducción. Y el O(1) que se multiplica es cuando se agrega un id de canción al hash y a la lista de 10 canciones. Por cómo funciona la notación big O. La complejidad final del requerimiento seria de:

$$O(n)$$

Análisis de rendimiento:

Para el análisis de rendimiento, se comprobó el uso de memoria y el tiempo de ejecución de los requerimientos en las máquinas de los dos estudiantes. Las características técnicas de dichas maquinas se presentan en la siguiente tabla:

	Máquina 1	**Máquina 2**
Procesadores	Intel(R) Core (TM) i7-10750H CPU @ 2.60GHz 2.59 GHz	AMD Ryzen 3 3200G with Radeon Vega Graphics 3.60 GHz
Memoria RAM (GB)	16.0 GB	16.0 GB
Sistema Operativo	Windows 10 Home Single Language	Windows 10 Pro

Para realizar los requerimientos se utilizó para cada requerimiento las siguientes entradas:

Requerimiento 1:

- Instrumentalness en un rango entre 0.75 a 1
- Acousticness en un rango entre 0.25 a 0.75

Requerimiento 2

- Spechness entre 0.75 a 1
- Liveness entre 0.5 a 0.75

Requerimiento 3:

- Valence entre 0.6 a 0.9
- Tempo entre 100 y 140

Requerimiento 4:

- Se busca los siguientes géneros:
    - Reggae

- Pop
- Rock

La máquina 1 es de la estudiante Ana Sofía Castellanos Mosquera y la maquina 2 es del estudiante Martín Santiago

Resultados Maquina 1:

Para la máquina uno se obtuvieron los siguientes resultados a partir del archivo con el 20% de los datos:

	Tiempo [ms]	Memoria [Kb]
Req 1	650.684	19.43
Req 2	12.82867	20.141
Req 3	7240.76	24.41333
Req 4	22148.34	25.49467

Al comparar los Requerimientos se observa que el que presenta menor consumo de memoria es el Requerimiento 1 y el menor tiempo el Requerimiento 2. La diferencia entre el 2 y el tres a pasar de que intersecan dos características específicas radica en que dentro del segundo ciclo el requerimiento 3 implementa de una vez la lista de retorno lo cual hace necesario, otra verificación del tamaño de la lista y añadirlo al final.

Resultados Maquina 2:

Los resultados de la maquina 2 se expresan mediante la siguiente tabla:

	Tiempo [ms]	**Memoria [kB]**
Req 1	829,08	23,36
Req 2	7,04	20,26
Req 3	9316,97	22,67
Req 4	22538,35	24,20

La forma en la que se obtuvieron estos resultados fue en que, se repitió el requerimiento 3 veces seguidas usando los mismos parámetros de búsqueda. Estos parámetros de búsqueda fueron los siguientes para los requerimientos:

Mas información acerca de los resultados individuales y graficas se encuentran en el archivo de Excel en Data-Maquina 1