Observaciones reto 3

José Vicente Vargas Panesso – 201815601

Daniel Reales – 201822265

Análisis de complejidades:

1. **<u>Requerimiento 1:</u>**
   Inicialmente en la implementación de este requerimiento se utiliza la función keys(), perteneciente al TAD, esta función retorna las llaves en determinado rango para la característica 1 ingresada, y dado que se esta implementando un RBT se puede decir que esta función es $O(\log_2 n)$. Ahora se hace un recorrido según la cantidad de llaves que cumplan el rango en la característica 1 (n_{c1}) . Posteriormente se hace otro recorrido sobre cada una de las llaves que se encuentren en el rango especificado para la característica 2, es decir un recorrido de (n_{c2}). Cabe resaltar que dentro de cada uno de estos recorridos se usa la función get(), propia del tag, la cual tiene complejidad $O(\log_2 n)$. Finalmente, la complejidad total del requerimiento seria:

   $$O((\log_2 n) + n_{c1} * (\log_2 n_{c2} + n_{c2} * (\log_2 n_{c2})))$$

   Lo cual se puede aproximar a una complejidad aproximada:

   $$O(\log n * \ n_{c1} * n_{c2})$$

   Esto indica claramente que la complejidad del algoritmo se ve claramente afectada por el tamaño del rango especificado, entre más grande sea este mayor será el tiempo de ejecución.

2. **<u>Requerimiento 2:</u>**
   La implementación de este requerimiento es muy parecida a la del requerimiento 1, en el sentido de que usan las mismas funciones para retornar las llaves entre un rango especifico, y luego se hace un recorrido por cada una de las características, que en este caso puntual son 'liveness' y 'speechiness'. Por lo cual la complejidad es la misma a la anterior.

   $$O((\log_2 n) + n_{c1} * (\log_2 n_{c1} + n_{c2} * (\log_2 n_{c2})))$$

   Lo cual se puede aproximar a una complejidad aproximada:

   $$O(\log n * \ n_{c1} * n_{c2})$$

3. **<u>Requerimiento 3:</u>**

   La implementación del requerimiento 3 es idéntica a la del requerimiento 2 salvo por la elección distinta de géneros que sirven para filtrar los eventos de reproducción. En primer lugar, la primera subrutina invocada es aquella referente a obtener los valores en el rango indicado de llaves para el árbol RBT creado que tiene como llaves el valor del primer criterio ("valence"). Esta tiene una complejidad de $O(\log_2(n))$. Una vez con esta lista compuesta de diccionarios (y cuya longitud denotaremos n_{c_1}) por cada elemento de la lista se procede a

inspeccionar el mapa guardado en la llave “tempo” del diccionario. Esto nos retorna un mapa ordenado al cual, una vez más se le aplica la subrutina para obtener los valores entre las llaves (parámetros de filtro) indicados. Como el anterior, esta subrutina tiene una complejidad de $O(\log_2(n))$.

Con estos dos filtros obtenemos una segunda lista de tamaño n_{c_2} que contiene listas donde se encuentran los elementos. Para poder obtener los elementos de utiliza la función get() del TAD ordered map. Esta tiene una complejidad de $O(\log_2(n))$ al ser un RBT. En resumen, se tiene que el orden el orden de crecimiento temporal está dado por

$$O(\log_2(n) + n_{c_1} * (\log_2 n_{c2} + n_{c2}))$$

Lo cual se puede aproximar a

$$O(n_{c1} n_{c2})$$

4. **Requerimiento 4:**

El requerimiento 4 realiza un ciclo sobre los géneros ingresados por parámetro creando un conjunto de listas que contienen los eventos de escucha cuyo tempo se encuentra en el intervalo del genero en cuestión. Para hacer esto, se itera sobre los géneros y se utiliza la subrutina values() del TAD ordered map para obtener la lista. Esto tiene una complejidad de $O(n_1 \log_2(n)))$ donde n_1 corresponde al número de géneros y n al número de eventos que cumplen con el tempo de cada uno de los géneros. Posteriormente, aún dentro del ciclo por cada género se itera sobre la lista retornada por la subrutina values. Dado que esta lista de retorno esta compuesta de más sublistas, se itera sobre ella para finalmente obtener la información de los eventos.

En relación con lo anterior vemos que la complejidad temporal puede ser descrita por:

$$O(n_1 (\log_2(n) + n)$$

Esto puede ser aproximado a

$$O(n_1 n)$$

	Máquina 1	**Máquina 2**
Procesadores	Intel core i7-7700HQ	Intel i5-8250U (8) @3.4 Ghz
Memoria RAM (GB)	16 GB	8 GB
Sistema Operativo	Windows 10	Arch Linux

Tabla 1. Especificaciones de las máquinas para ejecutar las pruebas de rendimiento.

Maquina 1

Resultados

Carga	Small	5pct	10pct
Tiempo (ms)	88813.431	1106796.843	2287764.865
Espacio (kB)	1512000.829	6085460.788	8905047.252

Tabla 1. Comparación de Desempeño en términos de tiempo y espacio para cada una de las implementaciones de la carga de los archivos

Requerimiento 1	Small	5pct	10pct
Tiempo (ms)	128.574	12821.8813	19481.100
Espacio (kB)	34.549	36.8076171875	34.495

Tabla 2. Comparación de Desempeño en términos de tiempo y espacio para cada una de las implementaciones del Requerimiento 1

Requerimiento 2	Small	5cpt	10pct
Tiempo (ms)	194.579	4911.242	8372.874
Espacio (kB)	23.435	28.615	28.896

Tabla 3. Comparación de Desempeño en términos de tiempo y espacio para cada una de las implementaciones del Requerimiento 2

Requerimiento 3	Small	5pct	10pct
Tiempo (ms)	390.000	5678.234	11122.4193
Espacio (kB)	14.25	25.023	26.570

Tabla 4. Comparación de Desempeño en términos de tiempo y espacio para cada una de las implementaciones del Requerimiento 3

Requerimiento 4	Small	5pct	10pct
Tiempo (ms)	46121.58	301500.805	439459.8056
Espacio (kB)	394.599	3484.58	7037.208

Tabla 5. Comparación de Desempeño en términos de tiempo y espacio para cada una de las implementaciones del Requerimiento 4

Maquina 2

Resultados

Carga	Small	5pct	10pct
Tiempo (ms)	81975.008	*Kill*	*Kill*
Espacio (kB)	1511998.623	*Kill*	*Kill*

Tabla 1. Comparación de Desempeño en términos de tiempo y espacio para cada una de las implementaciones de la carga de los archivos

Requerimiento 1	Small	5pct	10pct
Tiempo (ms)	114.941	*Kill*	*Kill*
Espacio (kB)	35.916	*Kill*	*Kill*

Tabla 3. Comparación de Desempeño en términos de tiempo y espacio para cada una de las implementaciones del Requerimiento 1

Requerimiento 2	Small	5cpt	10pct
Tiempo (ms)	162.817	*Kill*	*Kill*
Espacio (kB)	22.935	*Kill*	*Kill*

Tabla 3. Comparación de Desempeño en términos de tiempo y espacio para cada una de las implementaciones del Requerimiento 2

Requerimiento 3	Small	5pct	10pct
Tiempo (ms)	396.406	*Kill*	*Kill*
Espacio (kB)	16.781	*Kill*	*Kill*

Tabla 4. Comparación de Desempeño en términos de tiempo y espacio para cada una de las implementaciones del Requerimiento 3

Requerimiento 4	Small	5pct	10pct
Tiempo (ms)	25292.943	*Kill*	*Kill*
Espacio (kB)	395.115	*Kill*	*Kill*

Tabla 5. Comparación de Desempeño en términos de tiempo y espacio para cada una de las implementaciones del Requerimiento 4

***Nota:* *Kill* se refiere al error arrojado por Python al momento de realizar la carga de los datos debido a que se excedió el límite de recursos disponibles en memoria principal. Por consiguiente, no se pudo cargar la información para los archivos con el 5 porciento y el 10 por ciento de los datos.