j.montealegresQuniandes.edu.co
l.gomez1@uniandes.edu.co

Estudiante 1: Juan Montealegre Cod 202012723
Estudiante 2: Lina María Gómez Cod 201923531

RETO 2- YOUTUBE ANALYSIS SOCIAL REHASH

A continuación, en la tabla 1 se muestra la comparación entre el Reto 1 y Reto 2. Para cada requerimiento se utilizaron las funciones getime(), getmemory() y deltatime() que se componían de las funciones nativas tracemalloc y time.perf_counter(). Asimismo, se muestran los parámetros utilizados:

Req 1:

`Catalog: Videos-large.csv`

`Tag: Music`

`Size: 3`

`Pais: canada`

Req 2:

Pais: Canada

Req 3:

Tag: Music

Req 4:

Catalog: Videos-large.csv

Pais: Canada

Size: 3

Tag en específico: 2018

Tabla 1. Comparación entre memoria y tiempo entre el Reto 1 y Reto 2

	Reto 1		**Reto 2**	
	Tiempo [ms]	**Memoria [kb]**	**Tiempo [ms]**	**Memoria [kb]**
Req 1	76554.19	39.37	4412.20	34.47
Req 2	364531.76	56.92	3444.46	5.73
Req 3	371531.14	52.3	3673.60	3.41
Req 4	83224.34	37.24	4656.84	1.8320

j.montealegresQuniandes.edu.co
l.gomez1@uniandes.edu.co

Estudiante 1: Juan Montealegre Cod 202012723
Estudiante 2: Lina María Gómez Cod 201923531

Conclusión: Se vieron mejoras significativas en el reto 2. Implementar tablas de Hash reduce el tiempo de ejecución, por una gran diferencia, además de eso en el reto 1 solo usamos un catálogo el general lo cual género que los tiempos de búsqueda aumentaran, a diferencia en este reto 2 se usaron más catálogos para hacer una búsqueda más eficiente. Además de lo escrito anteriormente donde se presenta un cambio real es en el req2 y req 3, en donde anteriormente teníamos una complejidad de más de O(n) debido a que recorríamos la lista 3 veces con un for para filtrar los datos, para este reto al usar más catálogos y ordenar la lista con más funciones de comparación solo se tiene que recorrer una parte de la lista una vez bajando los tiempos de más de 3 minutos a una cuestión de segundos.