# Documento de análisis

## Reto 2-EDA-2021-10-Grupo 12

Estudiante A: Julian Castro del Valle - j.castrod@uniandes.edu.co - 202020847

Estudiante B: Tomás Otero - t.otero@uniandes.edu.co – 202021733

**Análisis de complejidad por requerimiento:**

- Requerimiento 1: O(N) ya que cuenta con un ciclo for para recorrer la lista de los valores del catálogo, el cual es un mapa.

- Requerimiento 2 (Estudiante A): O(2N) ya que cuenta con dos ciclos for, uno para recorrer la lista de los valores del catálogo, el cual es un mapa, y otro para recorrer la lista de la que sale la respuesta final. Igualmente se podría tomar como O(N).

- Requerimiento 2 (Estudiante A): O(2N) ya que cuenta con dos ciclos for, uno para recorrer la lista de los valores del catálogo, el cual es un mapa, y otro para recorrer la lista de la que sale la respuesta final. Igualmente se podría tomar como O(N).

- Requerimiento 4: O(N) ya que cuenta con un ciclo for para recorrer la lista de los valores del catálogo, el cual es un mapa.

Para los requerimientos 1 y 4 se utiliza el algoritmo merge sort que tiene complejidad O(N log N) en promedio, siendo el algoritmo más rápido de los vistos en clase.

Se utiliza la estructura de datos tipo arreglo (ARRAY_LIST), al ser la más óptima en este caso en cuanto a velocidad y eficacia para almacenar datos. Adicionalmente, se usa el mecanismo de colisión Linear Probing, ya que también es el más eficiente entre los 2 mecanismos estudiados en clase.

**Observaciones importantes:**

- La función videos_a_dias_trending cuenta con complejidad O(N) pero usa merge sort, que tiene, en promedio, complejidad O( N log N). Se usa para contar los días que fue trending un video, o sea que se usa para el requerimiento 2 y 3.

- Las funciones printResults(O(N)), printResultsLikes(O(N)) y printResult2 (O(1)) son auxiliares para que se muestre la información requerida en el view, no influyen en cómo se organizan/filtran los datos.

Con respecto al reto 1 las diferencias no fueron muy notables a simple vista, ya que lo único a priori es que los requerimientos 2 y 3 pasaron a ser 2N, que se puede tomar como N. Sin embargo, mirando los tiempos de ejecución y la memoria que utilizó cada requerimiento para ser llevado a cabo fueron bastante diferentes.

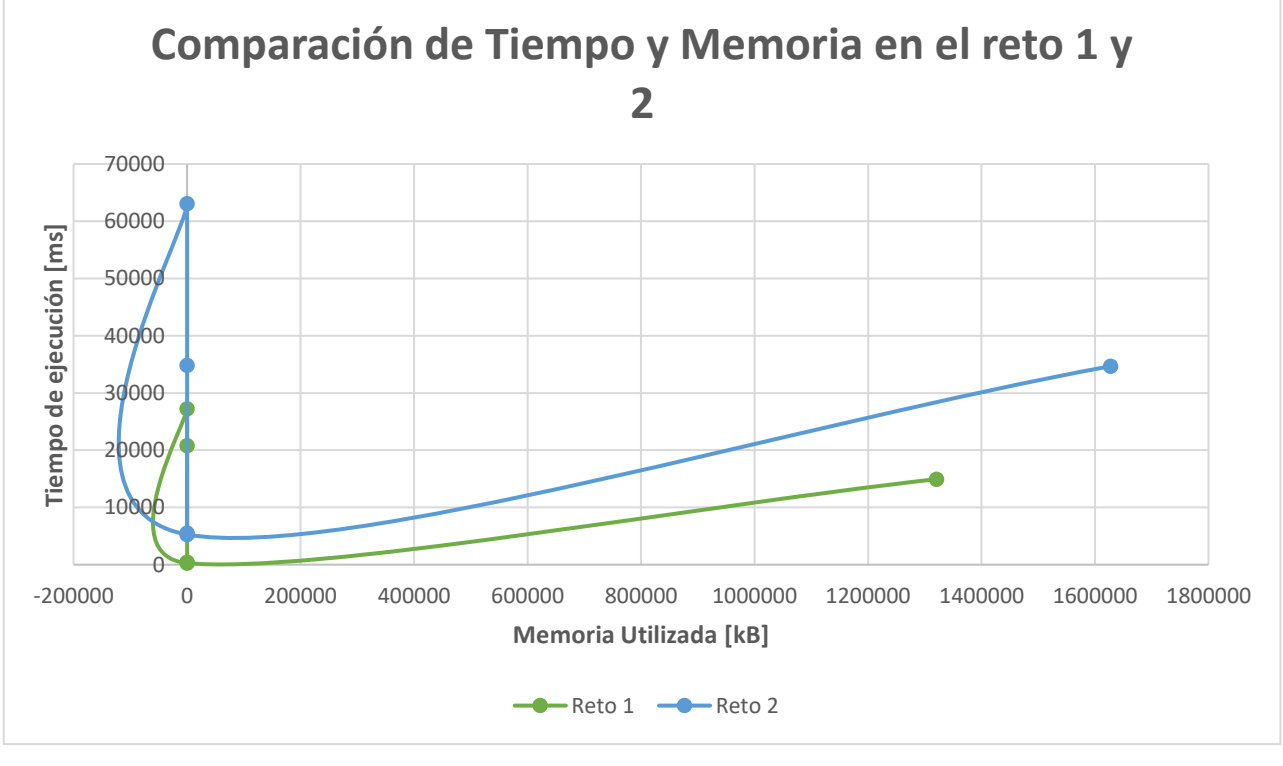
***Reto 1***

| Requerimiento | Consumo de Datos [kB] | Tiempo de Ejecución [ms] |
|---|---|---|
| 1.00 | 1320893.409 | 14936.406 |
| 2.00 | 52.802 | 274.219 |
| 3.00 | 19.824 | 27236.973 |
| 4.00 | 19.086 | 20817.641 |
| 5.00 | 71.349 | 450.412 |

***Reto 2***

| Requerimiento | Consumo de Datos [kB] | Tiempo de Ejecución [ms] |
|---|---|---|
| 1.00 | 1627406.461 | 34668.413 |
| 2.00 | 32.617 | 5262.75 |
| 3.00 | 29.024 | 63062.873 |
| 4.00 | 19.398 | 34816.515 |
| 5.00 | 66.852 | 5564.659 |

Como se puede ver en la tabla, el uso de memoria en el reto 1 fue menor que en el reto 2, a excepción del requerimiento 2 y 5, pero en términos generales fue mayor el uso de memoria en el reto 2. En cuanto al tiempo de ejecución, podemos ver que fue mayor en los requerimientos del segundo reto. A continuación, una gráfica para ilustrar mejor esto.



Como se puede evidenciar en la gráfica, el consumo de memoria y el tiempo de ejecución del reto 1 fue menor que en el reto 2. Se esperó que fuera lo contrario, siendo que los mapas son más eficaces que las listas para almacenar información. A pesar de que utilizamos Linear Probing, que según las pruebas realizadas en el laboratorio 7 fue más veloz que Separate Chaining, y que además en las listas utilizamos Array List, que en laboratorios anteriores demostró ser más eficaz que Linked List, no deja de llamar la atención el hecho de que se tomó menos tiempo y memoria en el reto 1 para llevar a cabo todos los requerimientos. A pesar de que no es una diferencia enorme de tiempo y gasto de memoria, es interesante ver que resultó siendo un poco más lento el reto 2.