**Análisis de Resultados**

**Reto 2 Grupo 4**

**Integrantes**

Ashlee Yin Romero, [a.yin@uniandes.edu.co](mailto:a.yin@uniandes.edu.co) 202421132

Daniel Galindo, [d.galindot@uniandes.edu.co](mailto:d.galindot@uniandes.edu.co), 202414673

Tomás Lozano, [t.lozanoc@uniandes.edu.co](mailto:t.lozanoc@uniandes.edu.co) 202422837

**Especificaciones**

La medición del tiempo de las funciones fueron tomadas en un computador MacBook Air 2015, viejita pero bien cuidada, con un procesador 1,6 GHz Dual-Core Intel Core i5 y memoria de 8 GB 1600 MHz DDR3. Todos los tiempos fueron tomados en milisegundos.

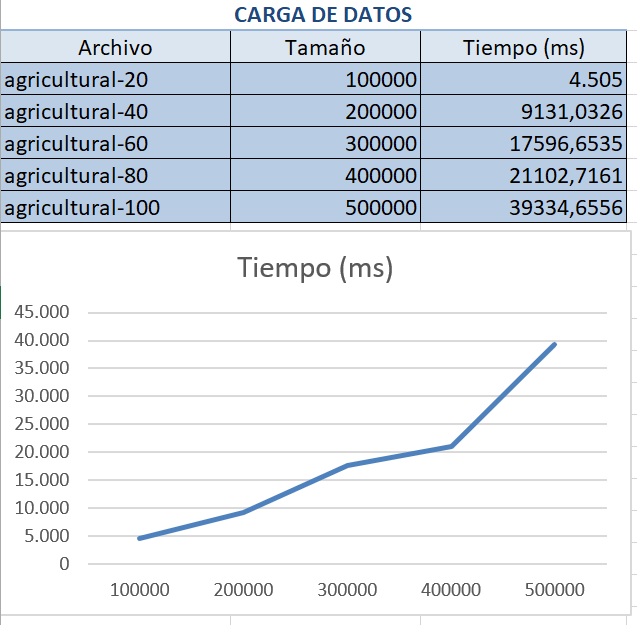
**Carga de datos**

La función de carga lee un archivo .csv con registros agrícolas y construye múltiples estructuras de almacenamiento utilizando mapas hash. Estas estructuras incluyen: un mapa general (registros), y otros mapas categorizados por año (por\_anio), departamento (por\_departamento), año y departamento (por\_anioydep), producto (por\_producto) y categoría estadística (por\_categoria).

Durante el proceso, la fecha de carga de cada registro es transformada al formato datetime. Posteriormente, los datos almacenados en cada estructura son ordenados utilizando el algoritmo merge\_sort.

* Este procedimiento tiene una complejidad algorítmica total de O(n log n):
* La lectura e inserción de registros en las estructuras tiene un costo lineal, O(n).
* El ordenamiento de las listas, mediante merge\_sort, tiene un costo de O(n log n).

Por lo tanto, al combinar ambos procesos, la carga completa de los datos mantiene una complejidad dominante de O(n log n).



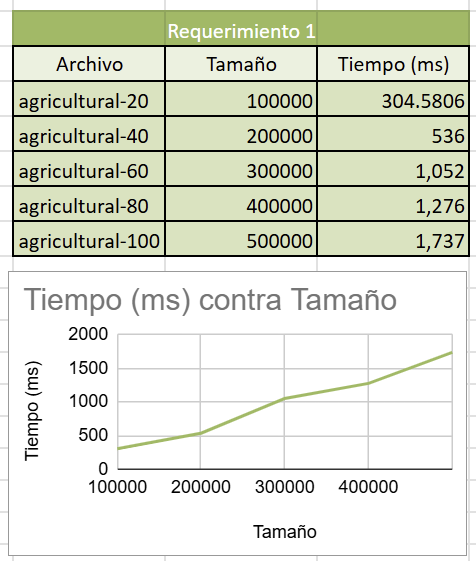
**Requerimiento 1 (Grupal)**

**Descripción:** Identificar el último registro recopilado en la plataforma para un año de interés específico. Este se determina utilizando el campo load\_time, que indica la fecha en que el dato fue cargado a la plataforma, y no necesariamente coincide con el año de recolección (year\_collection). El resultado incluye el total de registros encontrados para ese año y el registro con la fecha de carga más reciente.

**Análisis de Complejidad:** O(n)

**Parámetros:** 2007

**Pruebas de tiempo:**

****

**Requerimiento 2 (Grupal)**

**Descripción:**

Mostrar los últimos n registros cargados para un departamento específico.

No se realiza ordenamiento adicional, ya que los datos en el mapa catalog['por\_departamento'] ya se encuentran previamente ordenados por load\_time de forma ascendente al momento de la carga.

**Parámetros de entrada:**

**Departamento:** "CALIFORNIA"

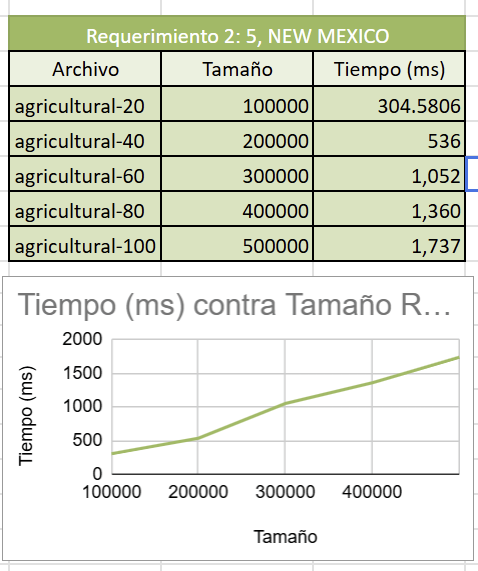
**Cantidad de registros n:** 10

**Análisis de Complejidad:**

Obtener la lista de registros por departamento desde el mapa tiene una complejidad O(1) (acceso directo tipo hash map).

Extraer los últimos n elementos con registros[-n:] es una operación de O(n) (donde n es el número solicitado, no la cantidad total).

Complejidad total: O(n), siendo n pequeño y fijo en la mayoría de los casos (por ejemplo, 10, 20…).



**Requerimiento 3 (Individual)**

**Descripción:**

Dado un departamento y un rango de años de recolección, identificar todos los registros que pertenecen a ese rango, junto con estadísticas sobre su origen (CENSUS o SURVEY) y ordenar los registros por load\_time (ascendente) y por state\_name (en caso de empate).

**Parámetros de entrada:**

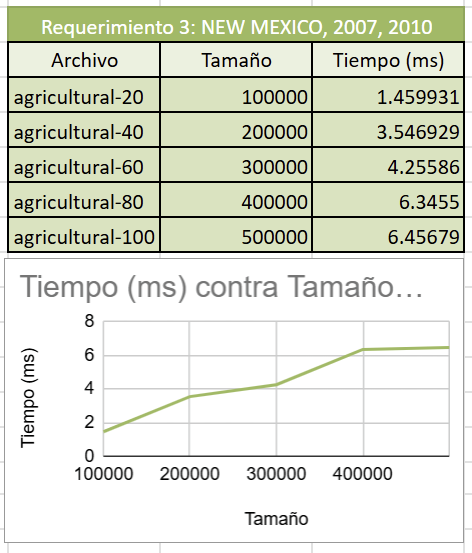
**Departamento:** "CALIFORNIA"

**Año inicial:** 2000

**Año final:** 2005

**Análisis de Complejidad Algorítmica:**

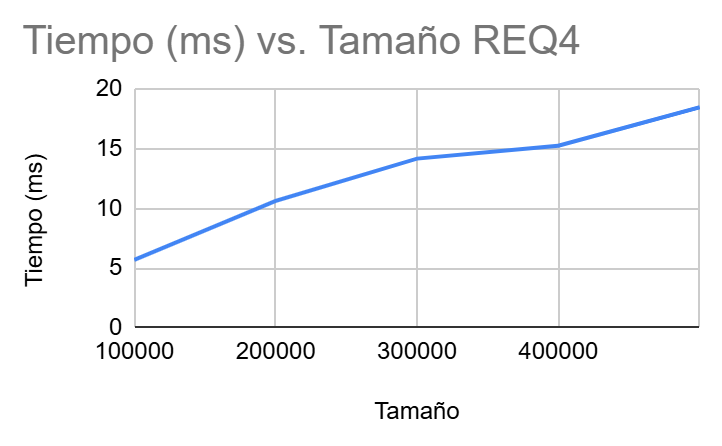
* Acceso al mapa catalog['por\_anioydep'] por departamento: O(1)
* Iteración por el rango de años (año\_inicial a año\_final): O(k), donde k = número de años en el rango.

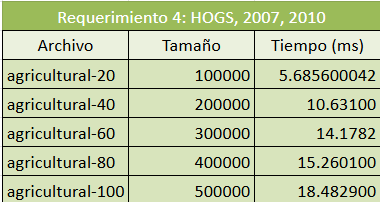


**Requerimiento 4 (Individual)**

**Descripción:**

Dado un producto y un rango de años, se buscan los registros asociados, se cuentan los de tipo SURVEY y CENSUS, y se presentan los 5 primeros y últimos registros según load\_time.

****

****

**Complejidad Algorítmica**

**Acceso al producto y años:** O(k), donde k es el número de años en el rango.

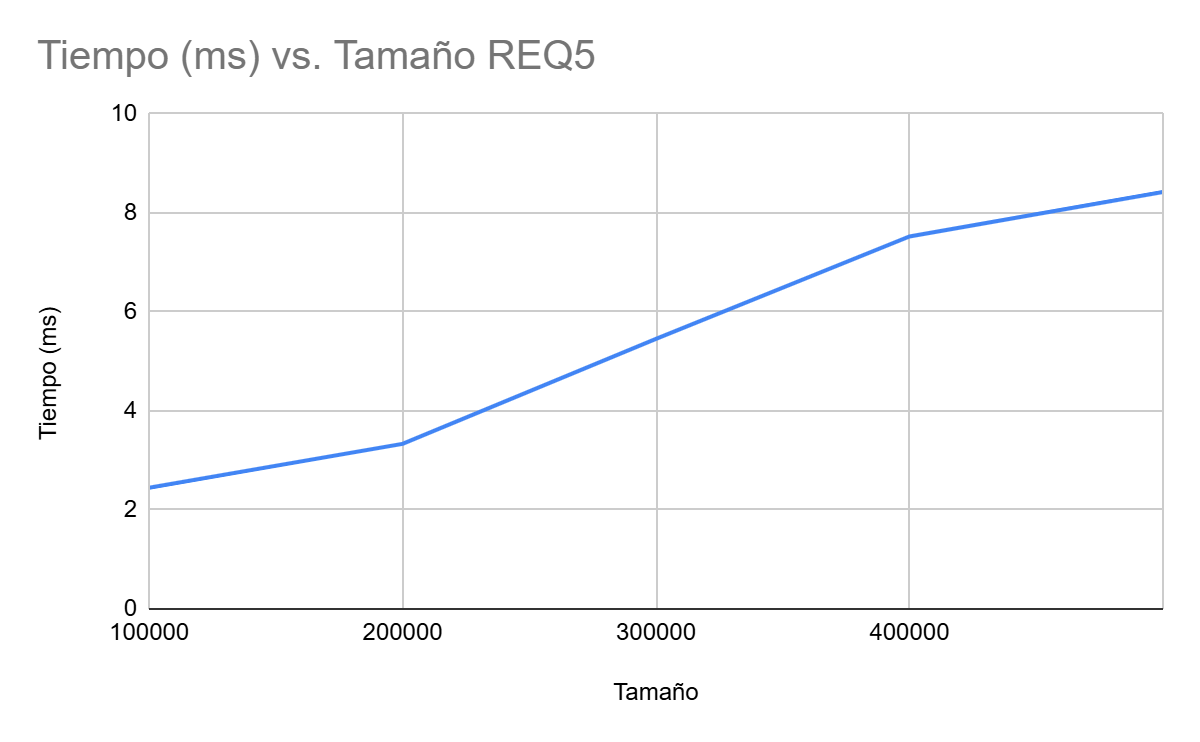
**Recorrido total de registros:** O(n)

**Ordenamiento por load\_time:** O(n log n)

**Total estimado:** O(n log n), siendo n los registros filtrados.

**Requerimiento 5 (Individual)**

**Descripción:** Listar los registros cargados según su categoría estadística para un rango de tiempo de interés.

****

****

**Complejidad Algorítmica**

**Acceso al producto y años: O(k),** donde k es el número de años en el rango.

**Recorrido total de registros:** O(n)

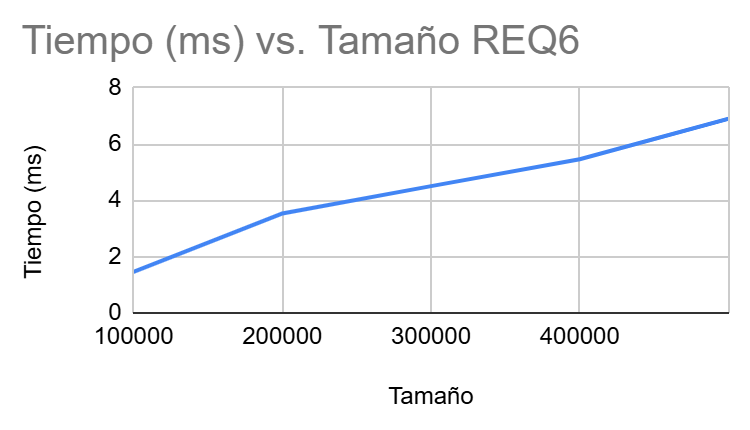
**Ordenamiento por load\_time:** O(n log n)

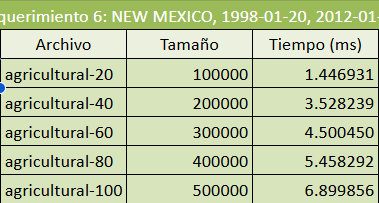
**Total estimado:** O(n log n), siendo n los registros filtrados.

**Requerimiento 6 (Grupal)**

**Descripción:**

Dado un departamento y un rango de fechas de carga, retorna todos los registros que cumplan esos filtros. Además, cuenta los registros con fuente SURVEY y CENSUS, y los ordena por load\_time descendente, usando state\_name como criterio secundario.

****

****

**Complejidad Algorítmica**

**Búsqueda del departamento en el mapa:** O(1)

**Recorrido de registros del departamento:** O(n)

**Filtrado por fechas + conteo de tipo de fuente:** O(n)

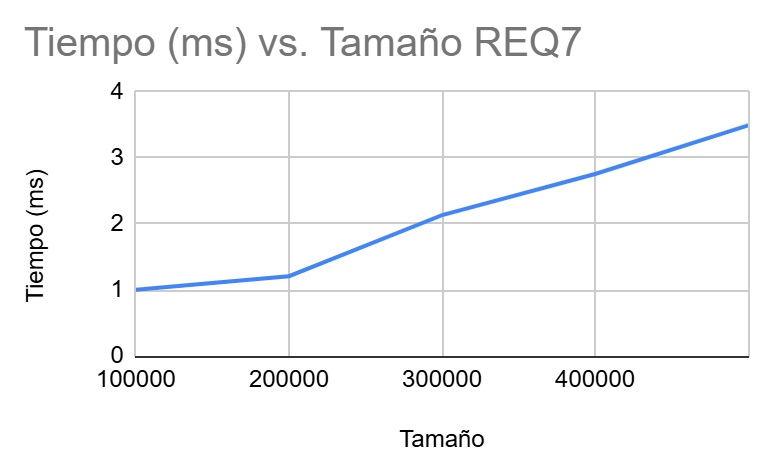
**Ordenamiento con merge sort:** O(n log n)

**Total:** O(n log n) donde n es el número de registros del departamento**.**

**Requerimiento 7**

**Descripción:**

Analizar el periodo con mejores o peores ingresos para un departamento para un rango de tiempo dado.

****

****

**Búsqueda del departamento en el mapa: O(1)**

**Recorrido de los años en el rango (inicial a final): O(n)**

**Acceso a registros por año (msc.get()): O(1)**

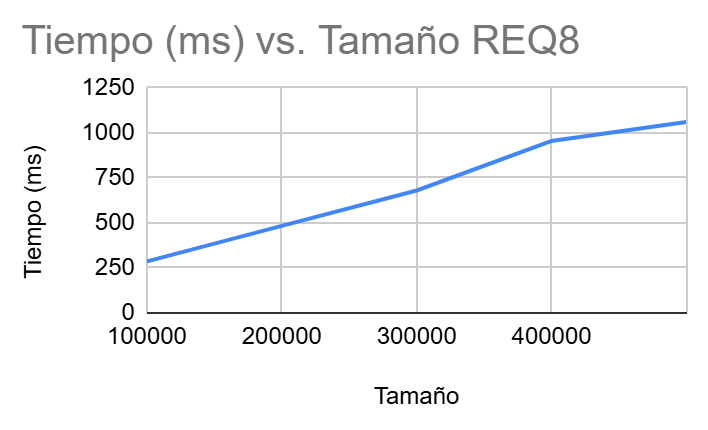
**Recorrido de registros por año: O(m)**

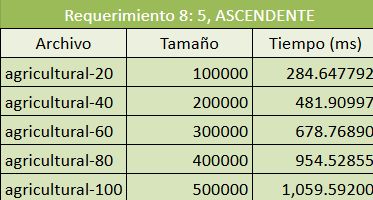
**Filtrado, conversión y conteo de valores: O(m)**

**Inserción en el diccionario resultados: O(n)**

**Ordenamiento con merge\_sort: O(n log n)**

**Bono Req 8**

****

****

**Obtener los departamentos (msc.key\_set()):** O(1)

**Recorrer los departamentos:** O(d) donde d es el número de departamentos

**Obtener registros de un departamento (msc.get()):** O(1)

**Recorrer registros dentro de un departamento:** O(m) donde m es el número promedio de registros por departamento

**Filtrar, comparar y acumular valores:** O(m)

**Agregar elementos a la lista (lt.add\_last()):** O(d)

**Ordenamiento con merge\_sort:** O(d log d)

Total: O(d \* m + d log d)