# **Observaciones del reto 2**

Ernesto Perez – 202112530 – ec.perez@uniandes.edu.co

Nicolás Saavedra – 20212963 - [n.saavedrag@uniandes.edu.co](mailto:n.saavedrag@uniandes.edu.co)

[**https://github.com/EDA2021-2-SEC01-G07/Reto3-G-07**](https://github.com/EDA2021-2-SEC01-G07/Reto3-G-07)

Requerimiento 3: Nicolás Saavedra

Requerimiento 4: Ernesto Perez

# **Análisis de complejidad**

# **Pruebas de rendimiento**

Especificaciones de la máquina de prueba

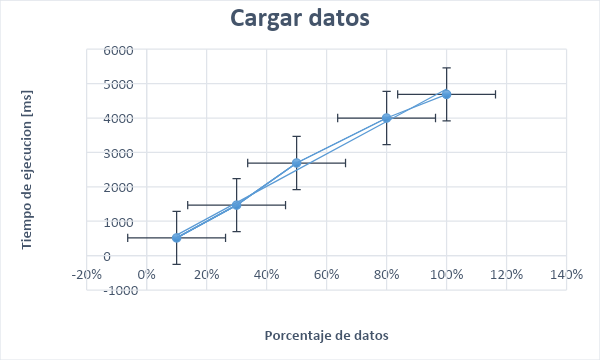
|  |  |
| --- | --- |
| Procesadores | AMD Ryzen 5 3600 6-Core Processor, 3593 Mhz, 6 Core(s), 12 Logical Processor(s) |
| Memoria RAM | 16GB |
| Sistema operativo | Microsoft Windows 10 pro 64-bits |

**Tiempo de ejecución promedio por requerimiento y crecimiento temporal**

|  | **Tiempo de ejecución para cada función [ms]** | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| **Porcentaje de datos** | **Cargar datos** | **Req1** | **Req2** | **Req3** | **Req4** | **Req5** |
| **10%** | 515.63 | 31.25 | 0.0 |  | 0.0 | 0.0 |
| **30%** | 1468.75 | 62.5 | 15.63 |  | 0.0 | 15.62 |
| **50%** | 2692.71 | 83.33 | 31.25 |  | 3.91 | 15.625 |
| **80%** | 4000 | 130.21 | 41.67 |  | 11.72 | 15.625 |
| **100%** | 4687.5 | 156.25 | 78.13 | 140.625 | 15.63 | 31.25 |

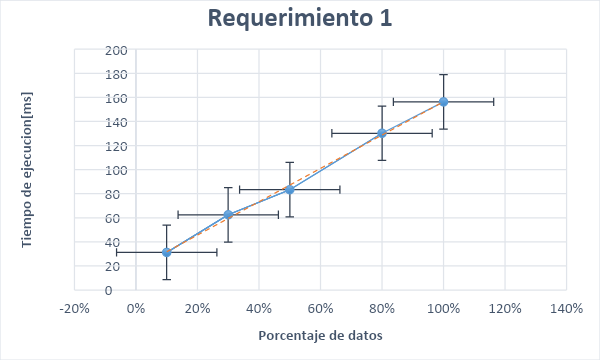
*En las funciones con entradas se usaron los mismos datos de entrada de los ejemplos.*

**Graficas de tiempo de ejecución promedio por requerimiento y crecimiento temporal**



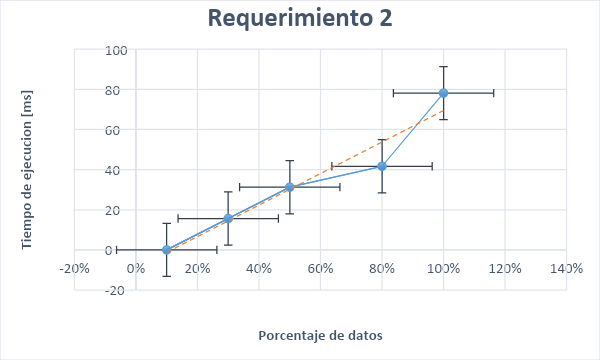
**Complejidad temporal carga de datos:** O (n)

Para cargar los datos se debe recorrer ambos archivos y por cada avistamiento meterlo a su respecto índice pero nunca se recorrer más de una vez el archivo.



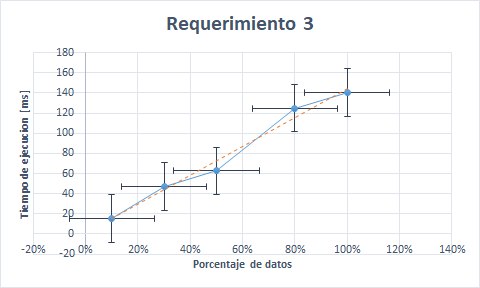
**Complejidad temporal requerimiento 1:** O (n log(n))

El requerimiento 1 tiene una complejidad lineal que se evidencia muy bien en el comportamiento de la gráfica. Esto se debe ya que se necesita conocer la ciudad con más avistamientos y la cantidad de avistamientos para esa ciudad por lo que ese necesario recorrer todo el árbol con índices de ciudades para poder conocer que tantos avistamientos hubo para dicha ciudad y de esta manera conocer cuál es la ciudad con mayor cantidad de avistamientos.

**v**

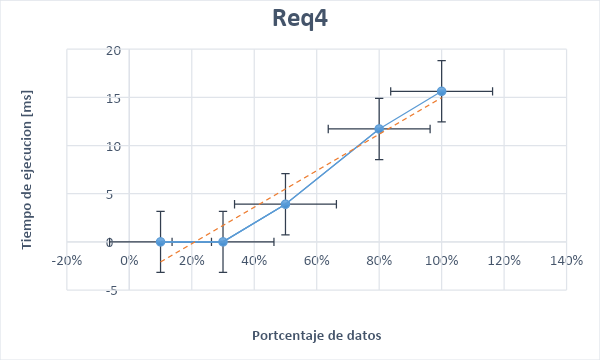
**Complejidad temporal requerimiento 2:** O(klog(k))

Debido a la naturaleza del requerimiento donde se requiere sortear por países y ciudades la complejidad temporal del requerimiento es igual a la de merge sort que es nlog(n) sin embargo, el sort más grande se realiza sobre un rango del árbol por lo que la complejidad no es n sino k, siendo k el tamaño del rango de duración que se esta consultando.



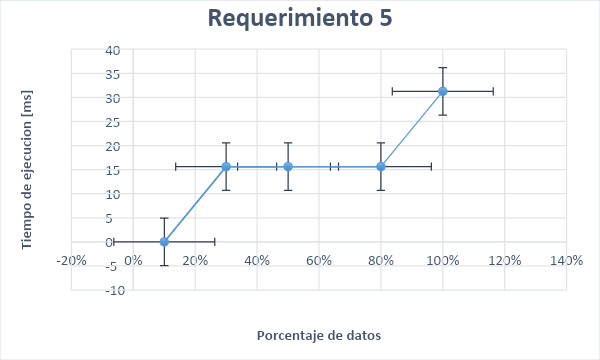
**Complejidad temporal requerimiento 3:** O(klog(h))

Ya que la operacion que mas computacion es el values para conseguir el rango de horas, se puede asumir que el rendimiento va a tomar este mismo rendimiento, ya que el resto de suboperaciones se hacen en sublistas con muy poco tamaño. k en este caso, seria el tamaño rango de valores tomados, siendo h la altura de el arbol con la cantidad de datos totales.



**Complejidad temporal requerimiento 4:** O(log(n))

Este requerimiento es más fácil ver en la gráfica un comportamiento lineal para la complejidad temporal. Esto se debe a que la función para sacar las llaves dado un rango tiene complejidad en el peor caso de O(n), una vez se saca este rango de valores el código es mucho más eficiente pues solo saca los 3 primeros y últimos en O (1).



**Complejidad temporal requerimiento 5:** O(k log(h))

Similar al requerimiento 3, la operacion mas costosa en este requerimiento termina siendo values, ya que la operacion para obtener los resultados que tienen una longitud adecuada dentro de la sublista es lineal, y el resto de suboperaciones se realizan en listas con tamaño negible para la complejidad.