Análisis del reto 3

Estudiante 1: Sergio Iván Rincón, 201914107, si.rincon@uniandes.edu.co
Estudiante 2: Luis Ernesto Tejón, 202113150, l.tejon@uniandes.edu.co

Noviembre de 2021

1. Observaciones generales

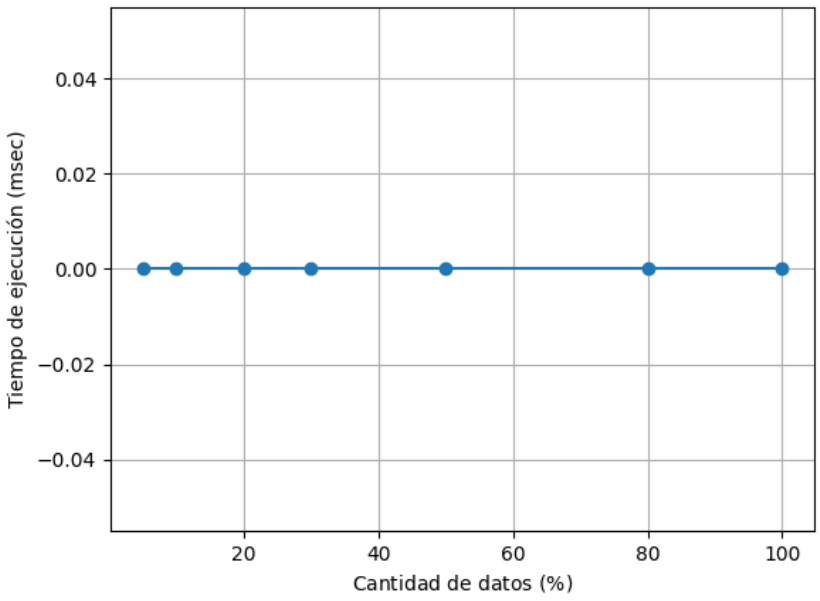
En este caso se buscó implementar los requerimientos del reto empleando árboles rojos negros principalmente, ya que esta estructura tiene un mejor rendimiento que los árboles binarios y no afectó significativamente los tiempos de carga de datos. De esta manera, se resolvieron los requerimientos empleando uno o varios arboles rojos negros dependiendo del caso.

Por último, como aclaración, los tiempos medidos en algunos de los requerimientos son de 0. Estos tiempos los medimos con módulo 'time' de Python del mismo modo a como se hizo en algunos de los laboratorios. Aunque no estamos seguros de por qué, pensamos que es posible que el módulo aproxime a 0 mediciones de tiempo extremadamente pequeñas. De lo que sí estamos seguros es que todas las consultas son bastante rápidas, lo que hace que no varíen significativamente entre el archivo small y large.

2. Análisis de complejidad por requerimiento

2.1. Requerimiento 1

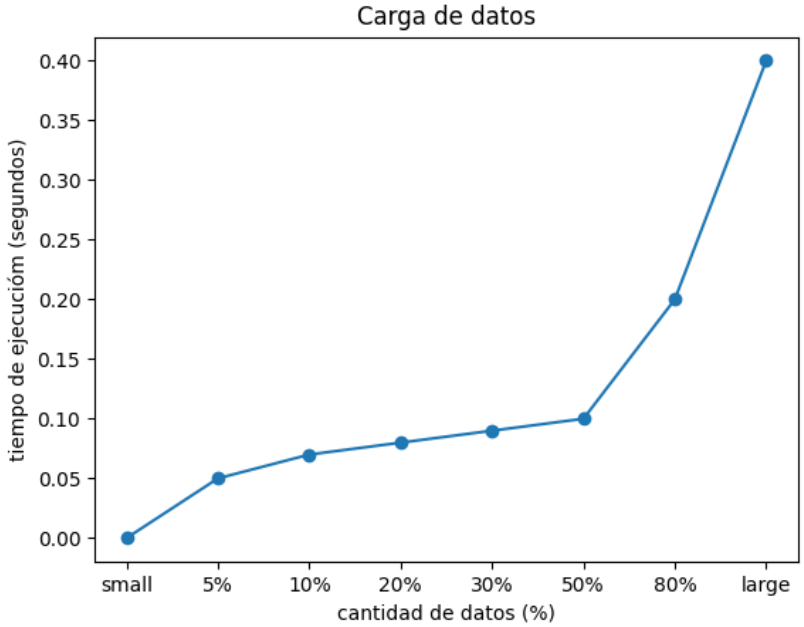
- **Tipo de TAD usado:** Mapa ordenado (arbol rojo negro) y mapa (con tratamiento de colisiones linear probing)
- **Carga de datos:** Para este requerimiento se crea un map con los nombres de las ciudades como llaves y con arboles rojos negros organizados por fechas como valores.
- **Descripción requerimiento:** Para solucionar este requerimiento primero se creó un map que permitiese organizar los avistamientos por ciudades. De esta manera, cada llave del map era el nombre de una ciudad y cada valor del map era un Arbol rojo negro asociado. En este caso se usó un RBT debido a que los avistamientos de la ciudad se debían listar por fechas. De esta manera, el arbol rojo negro de cada ciudad tiene como llaves las fechas de los avistamientos y como valores los datos especificos de cada avistamiento. Ahora bien, la complejidad de este requerimiento es $O(\log n)$ debido a que consultar el valor asociado a la ciudad es $O(1)$ y consultar los valores en el arbol rojo negro es $O(\log n)$
- Complejidad de la consulta: $O(\log n)$
- Gráfica de pruebas temporales (tiempos reportados en milisegundos y cantidad de datos en porcentaje):



2.2. Requerimiento 2 (implementado por Luis Tejón)

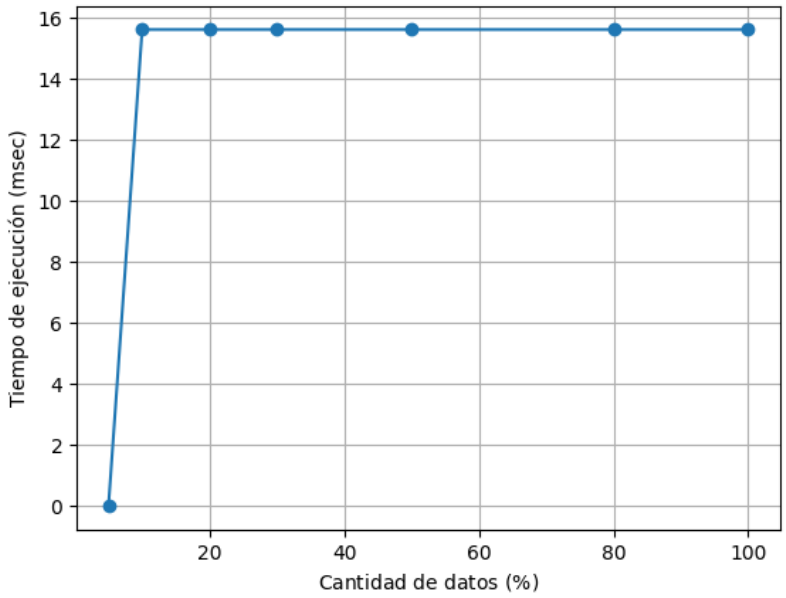
- **Tipo de TAD usado:** Mapa ordenado (RBT) y listas (ARRAY LIST).
- **Carga de datos:** Para este requerimiento en la carga de datos se crea un árbol RBT en el que los keys son las distintas duraciones en segundo de los avistamientos y los values una lista con los datos de los avistamiento que hayan tenido la duración de su respectivo key.

- **Descripción requerimiento:** Para encontrar la lista de avistamientos en el rango dado, basta con pasar la función values() del TAD árbol, y esta dará una lista (single linked) en la que cada valor es una de la listas de avistamientos en una duración, así que se itera sobre está lista de lista un lista final tipo Array que es la unión de todas las sublistas, por último se hace un merge de esta última lista para organizarla por duración, ciudad, país (en ese órden)
- **Complejidad de la consulta:** $O(nlogn)$, en el peor de los casos todos los avistamientos hacen parte del rango y por tanto habría que hacer merge de todos ellos, por tanto lo complejidad es de $O(nlogn)$, aunque en el caso promedio la complejidad sería $O(logn)$
- **Gráfica de pruebas temporales (en milisegundos):**



2.3. Requerimiento 3 (implementado por Sergio Rincón)

- Tipo de TAD usado: Map ordenado (RBT)
- Carga de datos: Para este requerimiento se crea un árbol rojo negro cuyas llaves son horas y cuyos valores son arboles rojos negros organizados por fecha y que contienen los avistamientos.
- Descripción requerimiento: En este caso se generó un arbol rojo negro cuyas llaves eran las horas de los avistamientos y valores eran otros árboles rojos negros. De esta manera, los arboles rojos negros asociados a cada hora tenían como llaves las distintas fechas y como valores los datos de cada avistamiento. Cabe aclarar que en este caso se revisó que no hubiese más de un avistamiento por fecha y hora específica, ya que en caso tal se hubiese tenido que implementar una lista de avistamientos como valor del segundo arbol rojo negro. A partir de lo anterior se puede decir que la complejidad de la consulta es $O(\log n)$ asociado a la obtención de las fechas multiplicada por $O(\log n)$ asociado a la obtención de las horas en cada fecha. De esta manera, la complejidad resultante es $O(\log^2 n)$
- Complejidad de la consulta: $O(\log^2 n)$
- Gráfica de pruebas temporales (en milisegundos):

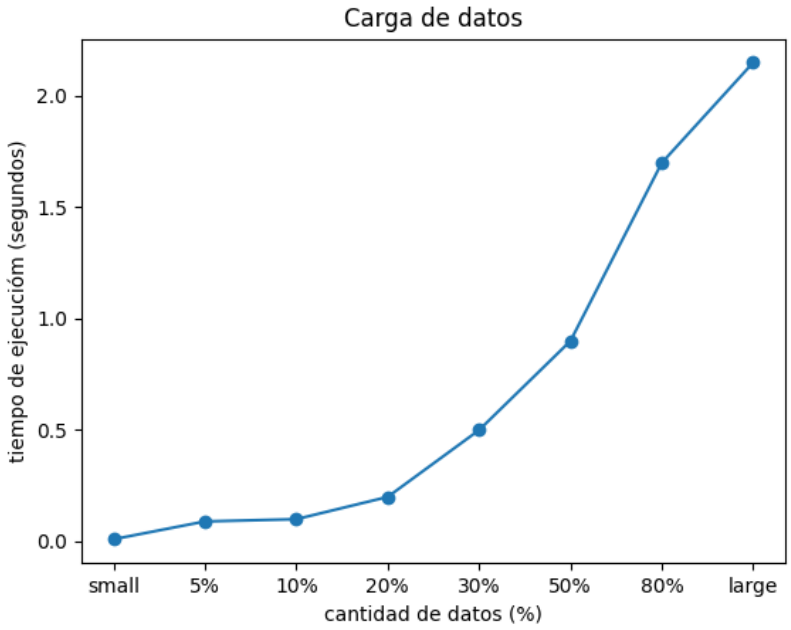


2.4. Requerimiento 4

- **Tipo de TAD usado:** Árbol (RBT) y lista (array)
- Carga de datos: En la carga de datos se hizo un árbol RBT en el que cada key era una fecha y hora distinta y el value era una lista (ARRAY LIST) con los avistamientos que hubiera ocurrido en esa fecha y hora.
- **Descripción requerimiento:** Se utiliza la función values(), para crear una lista (Single linked) de sublistas (array) las cuales contienen los avistamientos que estén entre el rango de

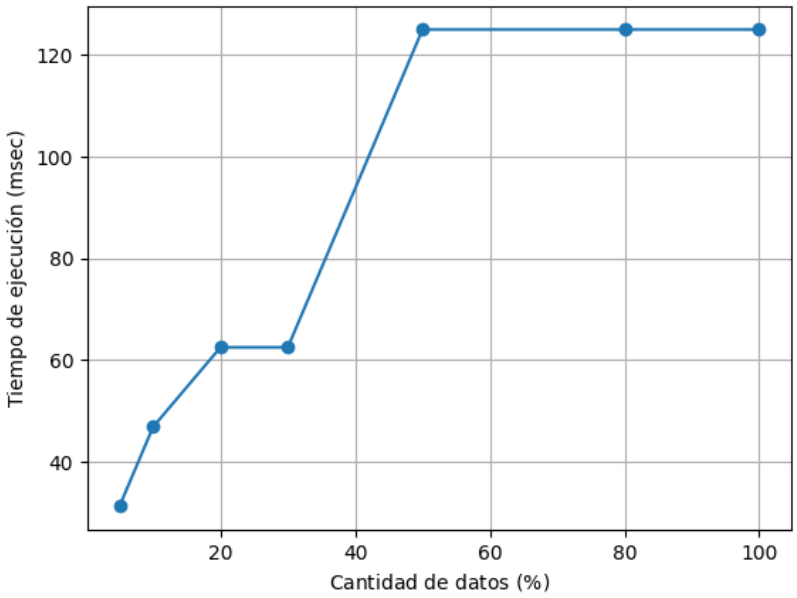
fechas dado, y luego se itera sobre esta lista para crear una lista grande (array) que contiene todos los avistamientos organizados cronológicamente en el rango dado.

- **Complejidad de la consulta:** Como la mayoría de los keys del árbol tiene solo un avistamiento en su lista, significa que al iterar sobre todas las listas para crear una gran lista se iterá casi que sobre los n avistamientos $O(n)$
- **Gráfica de pruebas temporales (en milisegundos):**



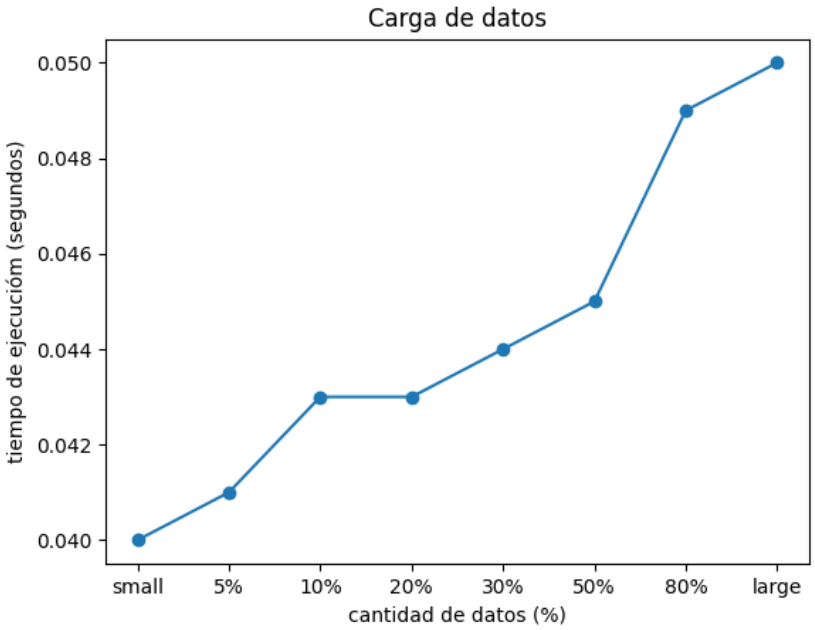
2.5. Requerimiento 5

- **Tipo de TAD usado:** Map ordenado (RBT)
- Carga de datos: Para éste requerimiento se genera un árbol rojo negro organizado por latitudes como llaves y con otros árboles rojos negros organizados por longitudes como valores.
- **Descripción requerimiento:** En este caso, se usó un arbol rojo negro organizado por latitudes de los avistamientos. De esta manera, cada arbol rojo negro tenía como llave la latitud y como valor otro arbol rojo negro organizado por longitud. A partir de esto, en la consulta se extraía una lista de arboles rojos negros organizados por longitud que se encontrasen en un rango de latitudes y, posteriormente, con esta lista se extraían sublistas de alistamientos entre las longitudes especificadas. Estas listas se unían finalmente y generaban la respuesta a la consulta. De esta manera la complejidad es $\log n \cdot \log n$, es decir $\log^2 n$
- **Complejidad de la consulta:** $O(\log^2 n)$
- **Gráfica de pruebas temporales (en milisegundos):**



2.6. Requerimiento 6 (bono)

- **Tipo de TAD usado:** Map ordenado (RBT)
- **Descripción requerimiento:** En este requerimiento simplemente se copia el requerimiento 5 pero se utiliza la librería folium para visualizar en el navegador el lugar de los avistamientos en un mapa del mundo.
- **Complejidad de la consulta:** $O(\log^2 n)$
- **Gráfica de pruebas temporales (en milisegundos):**



3. Datos adicionales

3.1. Ambientes de prueba

Los requerimiento 1, 3 y 5 se les midió el tiempo en la maquina 1, mientras que la carga de datos y los requerimientos 2, 4 y 6, se les midió el tiempo en la maquina 2.

	Máquina 1	**Máquina 2**
Procesadores	AMD Ryzen 5 3550H with Radeon Vega Mobile Gfx 2.10 GHz	Ryzen 5 5600g CPU @ 3.9GHz, 4.4GHz
Memoria RAM (GB)	16GB	12GB
Sistema Operativo	Windows 10 home 64-bits	Windows 11 home 64-bits

3.2. Tiempos de la carga de datos (en segundos):