OBSERVACIONES DEL RETO 3

Juan Pablo Rodríguez Briceño Cod 202022764

Nicolas Pérez Terán Cod 202116903

Ambientes de pruebas

	Máquina 1	**Máquina 2**
Procesadores	Chip M1	AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx 2.10 GHz
Memoria RAM (GB)	8 GB	12 GB (9,95 utilizables)
Sistema Operativo	MacOS BigSur	Windows 10 Home 64-bits

Maquina 1

Resultados

Porcentaje de la muestra [pct]	**Req – 1**	**Req - 2**	**Req - 3**	**Req - 4**	**Req - 5**	**Req - 6**
Small (768)	3.548 ms	9.2829 ms	47.0109 ms	10.7649 Ms	1.647 ms	27.0069 ms
5% (15008)	5.065 ms	38.4249ms	154.9239 ms	18.8369 Ms	4.141ms	58.969 ms
10%	7.5069ms	62.697ms	284.752ms	27.2890 Ms	6.8869 ms	84.0669 ms
20%	17.723ms	98.8419 ms	581.573ms	48.375 Ms	12.844 ms	133.197 ms
30%	16.719ms	127.5519 ms	886.776 ms	69.2840ms	19.0419 ms	178.6289 ms
50%	23.386 ms	197.8369 ms	1555.330ms	97.085Ms	30.2579 ms	274.8529 ms
80%	30.4140 ms	326.8360 ms	2590.9039 ms	133.845 ms	43.248 ms	517.6359 ms
100%	35.653ms	395.716 ms	3334.33 ms	187.2489 ms	49.889 ms	492.2459 ms

Maquina 2

Resultados

Porcentaje de la muestra [pct]	Req – 1	Req - 2	Req - 3	Req - 4	Req - 5	Req - 6
Small	15.625 ms	97.75 ms	62.5 Ms	15.625 ms	15.625 Ms	46.875 Ms
5%	15.625 ms	62.5 ms	328.125 Ms	15.625 Ms	15.625 Ms	78.125 Ms
10%	15.625 ms	78.125 ms	625.25 Ms	62.5 Ms	15.625 Ms	109.375Ms
20%	15.625 Ms	125 ms	1484.375 Ms	46.875 Ms	15.625 Ms	234.375Ms
30%	15.625 Ms	187.5 ms	2390.625 Ms	78.125 ms	15.625 Ms	437.5 Ms
50%	15.625 Ms	359.375 Ms	4437.5 Ms	125 Ms	31.25 Ms	578.125Ms
80%	31.25 Ms	578.125 Ms	7531.25 Ms	234.375 Ms	31.25 Ms	984.375Ms
100%	31.25 Ms	718.75 Ms	9812.5 Ms	265 Ms	46.875 Ms	1281.25Ms

Análisis de complejidad por cada requerimiento.

Requerimiento 1 (Grupal): Contar los avistamientos en una ciudad.

Este requerimiento se resuelve con *countCity(catalog['cityIndex'], city),* el cual tiene una complejidad de ~Log(N) + NLog(N) en el peor de los casos. Esto, ya que el map que se le da por parámetro es un RBT, y solo se necesita buscar una llave ('city', la cual se da como parámetro): esto tiene una complejidad de ~Log(N) y se hace llamando a la función *onlyMapValue(map, key),* la cual extrae y retorna el valor que se encuentra en esa llave. El valor extraído es una 'ARRAY_LIST', el cual se ordenara con un mergeSort, que tiene complejidad de NLog(N) en el peor de los casos.

Requerimiento 2 (Individual - Nicolas Perez Teran): Contar los avistamientos por duración

Este requerimiento se logra con dos funciones: *getDurRange(map, min, max)* y *getAllItems(lists).* El primero tendra una complejidad de N~log(N), ya que utiliza la *om.values(map,min,max),* el cual buscara en un arbol todos los elementos que estén en el rango [min, max] y los devolverá en forma de lista; esto teniendo que repetir ~Log(N) un total de N veces, en el peor de los casos. Luego, se llama a la funcion *getAllItems(lists),* que

será la encargada de recorrer la lista que devuelve *values(map, min, max),* entonces tendra una complejidad de N*N, ya que cada elemento de la lista es otra lista y obligatoriamente tiene que recorrer todos los elementos de cada una, sin embargo, nunca se devuelve.
La complejidad final es O(N^2 + N~Log(N))

Requerimiento 3 (Individual - Juan P. Rodríguez B.): Contar avistamientos por Hora/Minutos del día

Este requerimiento se resuelve con *countTime(catalog,timeMin,timeMax),* el cual tiene una complejidad de ~Log(N) + ~Log(N) + NLog(N) en el peor de los casos, ya que trabaja con los datos que se encuentren dentro de un RBT y que también se encuentre dentro del rango de hora que se haya establecido por parámetro. Primero, se extraen las llaves que cumplan la condición de estar en el rango de tiempo usando *keys(map,timeMin,timeMax)* con complejidad de N*N ya que funciona similar a *values(map,timeMin,timeMax).*
Posteriormente, se llama la función *onlyMapValue(map,key)* cuya complejidad de ~Log(N) y se llama por cada llave que exista en el rango. Luego de extraer los datos necesarios, se organizan con mergeSort, que cuenta con una complejidad de NLog(N) en el peor de los casos. La complejidad final es O(N^2 + 2~Log(N) + NLog(N))

Requerimiento 4 (Grupal): Contar los avistamientos en un rango de fechas

Este requerimiento se resuelve con *countTime(catalog,dateMin,dateMax),* el cual tiene una complejidad de ~Log(N) + ~Log(N) ya que trabaja con los datos que se encuentren dentro de un RBT y que también se encuentre dentro del rango de fechas que se haya establecido por parámetro. Primero, se extraen las llaves que cumplan la condición de estar en el rango de fechas usando *keys(map,dateMin,dateMax)* con complejidad de N*N como se explico anteriormente. Por último, se llama la función *onlyMapValue(map,key)* cuya complejidad de ~Log(N) y se llama por cada llave que exista en el rango. La complejidad final es (N^2+~Log(N))

Requerimiento 5 (Grupal): Contar los avistamientos de una Zona Geográfica
Nota: La sección del catálogo utilizada en esta parte es un RBT (Ordenado a partir de la longitud), que almacena en cada llave otro RBT (Ordenado a partir de la latitud), el cual a su vez en cada llave almacena una lista con los avistamientos.

Este requerimiento se divide en dos partes: *getLonRange(catalog, min, max)* y *getLatRange(list, minLat, maxLat).*
Primero, *getLonRange(catalog, min, max)* posee una complejidad de O(~Nlog(N)), porque tiene que buscar los elementos de un RT en un rango [min, max] y devolverlos en una lista; los datos para ser tomados tienen una complejidad de O(~Log(N)), y se repiten N veces, en el peor de los casos.
Segundo, *getLatRange(lst, min, max).* Que se divide en cuatro partes: la primera, tiene una complejidad de O(N), porque recorre toda la lista de mapas que se le dio por parámetro; la segunda, tiene complejidad de O(~NLog(N)) porque saca los valores de un mapa de listas, dado un rango de latitud [min, max] (Se utiliza *om.values(map, min, max),* cuya complejidad que ya se explicó anteriormente); la tercera, tiene complejidad de N*N, porque recorre cada elementos de la lista de listas y lo va añadiendo a una nueva lista, lo cual tiene complejidad de O(N) porque añadirá N elementos; y por último, hara un mergeSort, que tiene complejidad

de O(Nlog(N)).
La complejidad final sería de O(2N + ~2Nlog(N) + N^2 + NlogN)

Requerimiento 6 (BONO Grupal): Visualizar los avistamientos de una zona geográfica.

- Este requerimiento opera igual a como lo hace el requerimiento 5, el cual tiene una complejidad de O(2N + ~2Nlog(N) + N^2 + NlogN). Este requerimiento, en lugar de usar *agregarTabla(lista,int)* cuenta con una nueva función de nombre *mapSights(list,size,minLon,minLat,maxLon,maxLat)* la cual recorre los registros de los avistamientos en los rangos. Por cada registro, creará una tabla en HTML mostrando la ciudad, la fecha, la duración en segundos, la forma y los comentarios; una vez creada la tabla, la asignará al marcador del mapa en donde se haya avistado el OVNI. Por último, creará un cuadrado que representa una zona de acuerdo a los parámetros dados por el usuario y lo guardará todo en un archivo html.