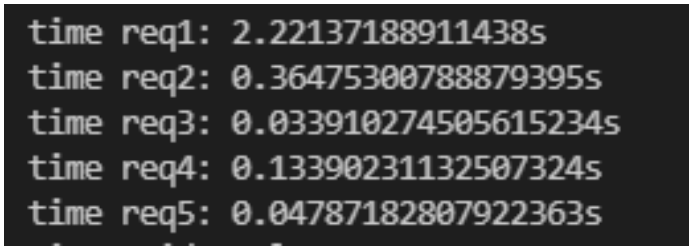
Link reto: https://github.com/EDA2021-2-SEC06-G07/Reto3-G07.git

Santiago Páez-202014644 Requerimiento 3

Daniel Barreto-201822639 Requerimiento 2

Tiempo

Para ejecutar los tiempos, debe seleccionar la opción 8.



- El requerimiento 1 se demora más porque estoy haciendo una serie de maromas entre poner todo en un mapa no ordenado mientras se llena y logro saber cuál es la ciudad con más avistamientos. El hecho de que no se pueda saber la ciudad con más avistamientos sin que el mapa este ordenado, hace que se tenga que hacer dos ordenamientos.
- El requerimiento 2 es más intuitivo, es solo meter todos los datos que cumplan una condición en un mapa y retornar el mapa. Por esto es que se demora menos.
- En el requerimiento, 3 el tiempo se redujo gracias a la creación de un árbol cuyas llaves son las horas cuyos valores son los avistamientos respectivos. Para realizar el requerimiento, solo sé hizo la búsqueda porque la creación del árbol es parte de la creación de datos.
- En el requerimiento, 4 el tiempo se redujo gracias a la creación de un árbol cuyas llaves son las fechas cuyos valores son los avistamientos respectivos. Para realizar el requerimiento, solo sé hizo la búsqueda porque la creación del árbol es parte de la creación de datos.
- En el requerimiento, 5 el tiempo se redujo gracias a la creación de un árbol cuyas llaves son las longitudes cuyos valores son los avistamientos respectivos. Para realizar el requerimiento, solo sé hizo la búsqueda porque la creación del árbol es parte de la creación de datos.
-

Complejidad

Req 1: Los principales movimientos de datos en este requerimiento son agregar todos los datos a un mapa de listas y luego mandar las listas de este a un arbol. El primer paso es orden O(n*m) donde n es el número de UFOs y m es el número de ciudades. Esto se debe a que, por cada insercion de un UFO, estamos mirando a ver si existe la ciudad dentro del mapa. La segunda parte es solo recorrer el mapa de ciudades y agregarlo a un arbol. Ese movimiento es O(mlogm) ya que por cada elemento se está haciendo una incercion en un árbol binario. Al final la complejidad es: O(mlogm + nm) donde n es el número de UFOs y m es el número de ciudades.

Req 2: O(nlogn) Ya que está recorriendo una lista con los valores y luego agrega cada uno a un arbol.

Req 3: O(nlogn) Porque realiza una la búsqueda de una llave y agrega su valor en una lista (El árbol fue creado en la carga de datos)

Req 4: O (nlogn) Porque realiza una la búsqueda de una llave y agrega su valor en una lista (El árbol fue creado en la carga de datos)

Req 5: O (nlogn) Porque realiza una la búsqueda de una llave y agrega su valor en una lista (El árbol fue creado en la carga de datos)