Camilo Ortiz Cruz 201821615 [c.ortizc@uniandes.edu.co](mailto:c.ortizc@uniandes.edu.co)

Kevin Fernando Gómez Camargo 202015120 [k.gomezc@uniandes.edu.co](mailto:k.gomezc@uniandes.edu.co)

**Maquinas**

|  |  |  |
| --- | --- | --- |
|  | **Kevin Fernando** | **Camilo** |
| **Procesador** | 2,5 GHz Intel Core i5 de dos  núcleos | 11th Gen Intel® Core™ i7-1165G7 @ 2.80Ghz |
| **Memoria RAM** | 4 GB | 16 GB |
| **Sistema Operativo** | macOS Catalina 10.15.7 | Windows 10 Home 64-bits |

**Pruebas de Tiempo:**

Los gráficos de las pruebas están al final del documento

**Camilo:**

**Imagen de la pantalla de un video juego

Descripción generada automáticamente con confianza baja**

Nota:

El tiempo total de cargar todos los archivos y correr todas las funciones 5 veces fue en promedio de 17s

**Requerimiento 1:**

Para poder realizar el requerimiento 1 se creo un RBT en el catalogo que tuviera como llave la ciudad y como valor una lista con todos los avistamientos en esa ciudad.

Texto

Descripción generada automáticamente

A partir del análisis realizado se llega a que la complejidad es O(n), consideramos que es una complejidad aceptable ya que toca revisar todas las ciudades para encontrar cual es la que más tiene por lo cual lo mínimo que se puede hacer es revisar todas las ciudades lo cual es n. Cabe resaltar que la complejidad del algoritmo será cuando el numero de avistamientos en una ciudad se mayor que el numero de ciudades en el árbol.

Al revisar las pruebas de tiempo se puede ver un crecimiento cuasi lineal pero ya que los tiempos son tan pequeños, el estado del computador puede afectar los tiempos.

**Requerimiento 2 Camilo:**

Para realizar el requerimiento 2 se creo un RBT que tenia como llaves las duración de un avistamiento y como valor una lista con los avistamientos:

Texto

Descripción generada automáticamente

La complejidad de este requerimiento consideramos que es muy buena ya que para solucionarlo es necesario encontrar los valores que están en el rango dado esto como mínimo será el numero de elementos que hay en el rango (k) por ende el algoritmo se acerca casi a lo mínimo posible para solucionar el requerimiento. SI vemos las pruebas de tiempo notaremos que es extremadamente rápido, básicamente toma 0 ms para cualquier tamaño de archivo.

**Requerimiento 4:**

Para realizar el requerimiento 4 se creo un RBT que tuviera como llave la fecha de un avistamiento y como valor una lista con los avistamientos.

Texto

Descripción generada automáticamente

Este algoritmo fue muy similar al requerimiento 2 pues los pasos para solucionarlos son casi idénticos cambiando solamente la llave sobre la que se quiere hacer el rango, consideramos que la complejidad es muy buena puesto que al igual que en el req. 2 se acerca casi a lo mínimo que se puede hacer para solucionarlo, al igual que en el 2 las pruebas de tiempo fueron de prácticamente 0 ms para todos los archivos, lo cual es muy bueno.

**Requerimiento 5:**

Para solucionar el requerimiento 5 se creo un RBT que tuviera como llave la latitud redondeada a 2 decimales y como valor tuviera un RBT con llave longitud y de valor una lista de los avistamiento que tuvieran esa latitud y longitud redondeada.

Diagrama estructura árbol:

Diagrama

Descripción generada automáticamente

Texto

Descripción generada automáticamente



La complejidad temporal de este algoritmo es más difícil de calcular ya que depende de diferentes elementos y estos son dependientes entre ellos, es por eso que no se simplifica mas ya que no hay forma de saber si log(c) el cual es el numero de elementos en el árbol de longitud es menor o mayor al numero de avistamiento en el rango de longitud dado (e\*p). Adicionalmente los valores de c y e\*p no son los mismos pues depende en que nodo de latitud estemos, no obstante, a partir de las pruebas y análisis de los datos se puede asumir que los valores de log(c) y e\*p son pequeños, en base al archivo large los valores son:

Texto

Descripción generada automáticamente

En base a esto podemos decir que la complejidad final es de pero en general variaría mucho dependiendo de los parámetros pasados, sin embargo, en las pruebas de tiempo se puede ver que incluso en el archivo large los tiempos de ejecución fueron también de casi 0

Para el análisis de los parámetros se usó el siguiente código:

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

**Requerimiento 6:**

Para el requerimiento 6 se reutilizó el código del requerimiento 5 por lo cual la complejidad será de lo cual según el análisis en promedio será , no obstante algo importante a resaltar es que debido a que se esta utilizando folium para crear el mapa esto agregar una constante al tiempo de ejecución y dado que no se conoce la implementación no se puede decir como afecta pero basado en las pruebas de tiempo es un incremento significativo pues los tiempos pasaron de estar entre 0 y 3 ms a entre 15 y 25 ms, esto nos da a entender que plantear los puntos toma un tiempo significativo.

**Gráficos:**

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Imagen que contiene Gráfico de líneas

Descripción generada automáticamente

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Imagen que contiene Forma

Descripción generada automáticamente

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Nota:

Dado que las funciones corren tan rápido en algunos casos el análisis grafico no es muy productivo puesto que el computador en casi 0 ms y los cambios que se dan (3ms) pueden deberse al estado del computador en el momento de las pruebas,