Documento de análisis

Req 2: Valeria Caro Ramírez – v.caro@uniandes.edu.co
Req 3: Sofia Velasquez Marin – s.velasquezm2@uniandes.edu.co

En este documento se registrará el tiempo de carga de los requerimientos y la carga de datos del Reto 3. Además, analizaremos la complejidad de cada uno de los requerimientos y adjuntaremos la gráfica del tiempo de ejecución de cada uno de estos utilizando los diferentes tamaños de los datos de los avistamientos de los UFOS.

- **Carga de Datos**

Tamaño	Tiempo de Ejecución [ms]
Small	125.0
5pct	656.25
10pct	1296.875
20pct	2468.75
30pct	4140.625
50pct	6984.375
80pct	11625.0
Large	14171.875

- **Requerimiento 1**

Variables importantes:

- n = número de elementos en el RBT de ciudades
- $\log_2(n)$= altura del RBT de ciudades

Análisis: Para el requerimiento 1 hicimos dos funciones. En la primera recorremos el índice de ciudades utilizando el método keySet() que tiene una complejidad O ($\log_2(n)$). Luego iteramos por las llaves que hay en el índice y esto tiene una complejidad de O(n). Luego obtenemos por cada llave su valor con el método get() que tiene una complejidad de O ($\log_2(n)$), como este método get() se hace por cada llave, decimos que la complejidad total de este método O($n * \log_2(n)$). En la segunda función, se busca la ciudad ingresada por parámetro con el método get(), se comprueba que esa ciudad existe, si existe se retorna la lista de avistamientos de la ciudad y sino, se retorna None. Nota*:* $log_2(n) < n$.

Complejidad: O($n + n * \log_2(n)$)

Tamaño	Tiempo de Ejecución [ms]
Small	0.0
5pct	15.625

10pct	46.875
20pct	62.5
30pct	93.75
50pct	140.625
80pct	203.125
Large	328.125

- **Requerimiento 2** (Valeria Caro)

Variables importantes:

- n = número de elementos en el RBT de la duración en segundos
- $\log_2(n)$= altura del RBT de la duración
- #llaves = número de llaves en el rango dado
- #avistamientos = número de avistamientos en rango de la duración dada

Análisis: Para el requerimiento 2 se utilizaron dos funciones. La primera, crea una lista, y saca los valores de las llaves en el rango de la duración dada con la función values() que tiene una complejidad de O$(\log_2(n) + \#llaves)$; se recorren esas llaves lo que tiene una complejidad O$(\#llaves)$ y por cada valor se recorre la lista de avistamientos en esa duración, lo que tiene una complejidad total de O$(\#avistamientos)$. La segunda función, retorna el top 5 por mayor duración usando el método get () y el método maxKey() los cuales tienen una complejidad O$(\log_2(n))$.

Complejidad: O$(\log_2(n) + \#llaves + \#avistamientos)$

Tamaño	Tiempo de Ejecución [ms]
Small	0.0
5pct	15.625
10pct	15.625
20pct	15.625
30pct	15.625
50pct	15.625
80pct	15.625
Large	15.625

- **Requerimiento 3** (Sofia Velasquez)

Variables importantes:

- n = número de elementos en el RBT de tiempo [HH:MM]
- $\log_2(n)$= altura del RBT de tiempo
- #llaves = número de llaves en el rango dado
- #avistamientos = número de avistamientos en rango de tiempo dado

Análisis: Para el requerimiento 3 se utilizaron dos funciones. La primera, crea una lista, y saca los valores de las llaves en el rango de tiempo dado con la función values () que tiene una complejidad de O($\log_2(n) + \#llaves$) ; se recorren esas llaves lo que tiene una complejidad O($\#llaves$) y por cada valor se recorre la lista de avistamientos de ese tiempo, lo que tiene una complejidad total de O($\#avistamientos$). La segunda función, retorna el top 5 horas más tardías usando el método get () y el método maxKey () los cuales tienen una complejidad O($\log_2(n)$).

Complejidad: O($\log_2(n) + \#llaves + \#avistamientos$)

Tamaño	Tiempo de Ejecución [ms]
Small	0.0
5pct	15.625
10pct	15.625
20pct	15.625
30pct	15.625
50pct	15.625
80pct	15.625
Large	15.625

- **Requerimiento 4**

Variables importantes:

- n = número de elementos en el RBT de la fecha [AAAA-MM-DD]
- $\log_2(n)$= altura del RBT de la fecha
- #llaves = número de llaves en el rango dado
- #avistamientos = número de avistamientos en rango de la fecha dada

Análisis: Para el requerimiento 4 se utilizaron dos funciones. La primera, crea una lista, y saca los valores de las llaves en el rango de la fecha dada con la función values () que tiene una complejidad de O($\log_2(n) + \#llaves$) ; se recorren esas llaves lo que tiene una complejidad O($\#llaves$) y por cada valor se recorre la lista de avistamientos en esa fecha, lo que tiene una complejidad total de O($\#avistamientos$). La segunda función, retorna el top 5 fechas más antiguas usando el método get() y el método minKey() los cuales tienen una complejidad O($\log_2(n)$).

Complejidad: O($\log_2(n) + \#llaves + \#avistamientos$)

Tamaño	Tiempo de Ejecución [ms]
Small	0.0
5pct	15.625
10pct	15.625
20pct	15.625

30pct	15.625
50pct	15.625
80pct	15.625
Large	15.625

- **Requerimiento 5**

Variables importantes:

- a = número de elementos en el RBT de latitud
- $\log_2(a)$= altura del RBT de latitud
- o = número de elementos en el RBT de longitud
- $\log_2(o)$= altura del RBT de longitud
- #llaves(a) = número de llaves en el rango de latitud dado
- #llaves(o) = número de llaves en el rango de longitud dado
- #avistamientos = número de avistamientos en rango de la fecha dada

Análisis: Para el requerimiento 5 se utilizó una función la cual saca los valores de las llaves de latitud dado con la función values() que tiene una complejidad de O($\log_2(a) + \#llaves(a)$); se recorren esas llaves lo que tiene una complejidad O($\#llaves(a)$) y por cada valor se sacan los valores de las llaves de longitud en el rango dado, con la función values(), que tiene una complejidad de O($\log_2(o) + \#llaves(o)$), pero los RBT de longitud tienen un tamaño mucho menor que los RBT de latitud, por ello no vamos a tomar en cuenta su complejidad. Se recorren esas llaves lo que tiene una complejidad O($\#llaves(o)$); finalmente se recorre la lista de avistamientos de cada llave de longitud, lo que tiene una complejidad total de O($\#avistamientos$).

Complejidad: O($\#avistamientos + \log_2(a) + \#llaves(a)$)

Tamaño	Tiempo de Ejecución [ms]
Small	0.0
5pct	15.625
10pct	15.625
20pct	15.625
30pct	15.625
50pct	15.625
80pct	15.625
Large	15.625

- **Requerimiento 6**

Variables importantes:

- n = número de avistamientos a mostrar

Análisis: Para el requerimiento 6 se utiliza la librería de pandas para crear un dataframe de la lista de avistamientos retornados en el requerimiento 5 (primeros y últimos 5 avistamientos en el área), para esto se crea un csv que guarda dicha información. Se creó una función que recibe los parámetros ingresados (latitud y longitud máxima y mínima) en el requerimiento 5, la cual crea un mapa de folium con dichos parámetros, posteriormente se utiliza el método apply() de pandas para aplicar una funcion al dataframe y se crea una función anónima (lambda), la cual funciona como un for loop, para recorrer las filas del dataframe, luego se crea un marcador por cada fila y se añade al mapa, utilizando el método folium.marker(), teniendo en cuenta la longitud y la latitud de cada avistamiento registrado. Nota: $n \leq 10$

Complejidad: O(1)

Tamaño	Tiempo de Ejecución [ms]
Small	125.0
5pct	125.0
10pct	125.0
20pct	125.0
30pct	125.0
50pct	125.0
80pct	125.0
Large	125.0