Análisis del Reto

Grupo 2 Sección 4

Requerimiento 3 - Daniel Rodriguez - ds.rodriguezf1@uniandes.edu.co - 202014760
Requerimiento 4 - Santiago Forero - s.forerog2@uniandes.edu.co - 202111446

N = UFOS.csv

Requerimiento 1

- Análisis complejidad temporal:

  M = Cantidad de llaves de Ciudades

  P = Cantidad de avistamientos en una ciudad en una fecha específica en promedio

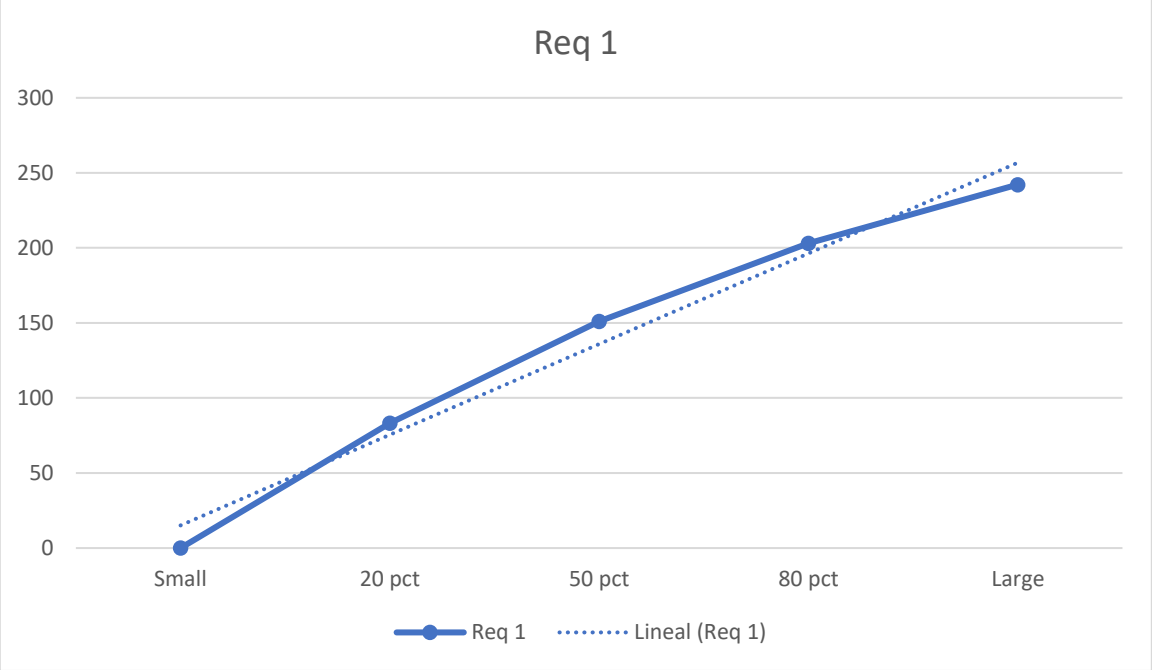
  La función contarAvistamientosCiudad tiene una complejidad temporal de O(m). Inicialmente obtiene la cantidad de ciudades con la función mp.size sobre el mapa no ordenado de ciudades (ya que las ciudades no tienen un criterio de ordenamiento entre sí), lo cual tiene una complejidad de O(1). Posteriormente recorre cada una de las llaves Ciudad con el fin de encontrar la ciudad con más avistamientos, lo cual tiene una complejidad O(m). Finalmente, dentro de la ciudad ingresada por el usuario obtiene el mapa ordenado por fechas con O (1), del cual extrae sus primeros y últimos tres avistamientos gracias a que es un mapa ordenado. En donde si llegan a haber varios avistamientos en un mismo día, los ordena por hora con Merge Sort O (p log(p)), sin embargo, esta complejidad es muy pequeña al no haber muchos avistamientos por ciudad y por fecha, además de solo necesitarse extraer 3 avistamientos antes de romper el ciclo. Así que la complejidad temporal del primer requerimiento es:

  O(m)

- Pruebas de tiempo – Tablas

	Small	20 pct	50 pct	80 pct	Large
Req 1	0	83.34	151.04	203.125	242.19

- Gráfica



Se puede ver que tiende a ser lineal, un poco curva debido a que los intervalos que existen entre los tamaños de carga no son iguales. También se puede ver que esta función se demora más que al requerimiento 3 o 4 debido a que hay más ciudades que fechas u horas específicas.

Requerimiento 2 – Daniel Rodriguez

- Análisis complejidad temporal:
- Pruebas de tiempo – Tablas

	Small	20 pct	50 pct	80 pct	Large
Req 2					

- Gráficas

Requerimiento 3 – Santiago Forero

- Análisis complejidad temporal:

  M = Cantidad de llaves Horas dentro del rango Ingresado por el usuario

  P = Cantidad de avistamientos en una hora específica en promedio
- La función contarAvistamientosHora tiene una complejidad temporal de O(m). Inicialmente se obtiene la última llave del mapa ordenado por horas para obtener la hora con avistamientos más tardíos, de donde se saca la cantidad de avistamientos a esa hora con O (1). Posteriormente se usan la función dt.time, om.ceiling, om.floor y om.values para

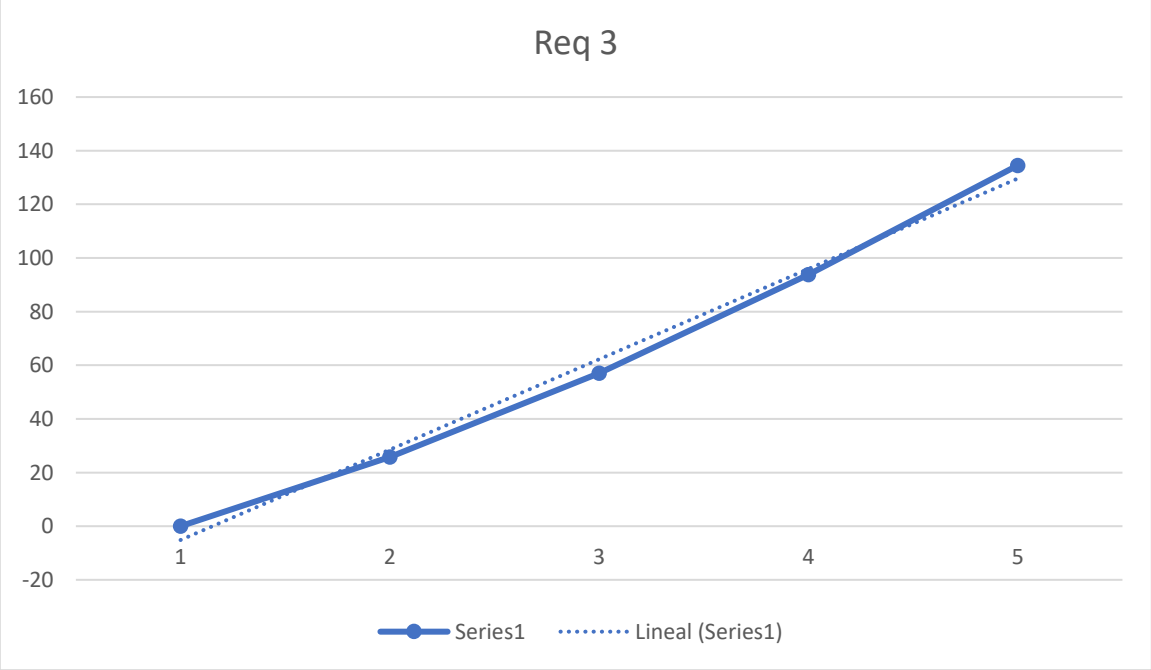
obtener los valores en el rango de horas ingresado por el usuario con O(1), dentro del cual se recorre cada una y se va sumando la cantidad de avistamientos que tenga como valor cada llave hora con lt.size con complejidad O(m). Finalmente extrae sus primeros y últimos tres avistamientos por hora gracias a que es un mapa ordenado, en donde si llegan a haber varios avistamientos en una misma hora, los ordena por fecha con Merge Sort O (p log(p)), sin embargo, esta complejidad es muy pequeña al no haber muchos avistamientos por hora en específico y al solo necesitarse extraer 3 avistamientos antes de romper el ciclo. Así que la complejidad temporal del tercer requerimiento es:

O(m)

- Pruebas de tiempo – Tablas

	Small	20 pct	50 pct	80 pct	Large
Req 3	0	25.84	57.08	93.75	134.53

- Gráficas



Se puede ver que tiende a ser lineal, lo cual coincide con la esperado, los valores son menores que en el requerimiento 1 pero mayores que en el 4, esto debido a que aunque hayan menos horas específicas que fechas específicas, las funciones de ordenamiento p manejan más avistamientos, ya que hay mas avistamientos por hora en específico que por fecha en específico.

Requerimiento 4

- Análisis complejidad temporal:

  M = Cantidad de llaves Fechas dentro del rango Ingresado por el usuario

  P = Cantidad de avistamientos en una fecha específica en promedio

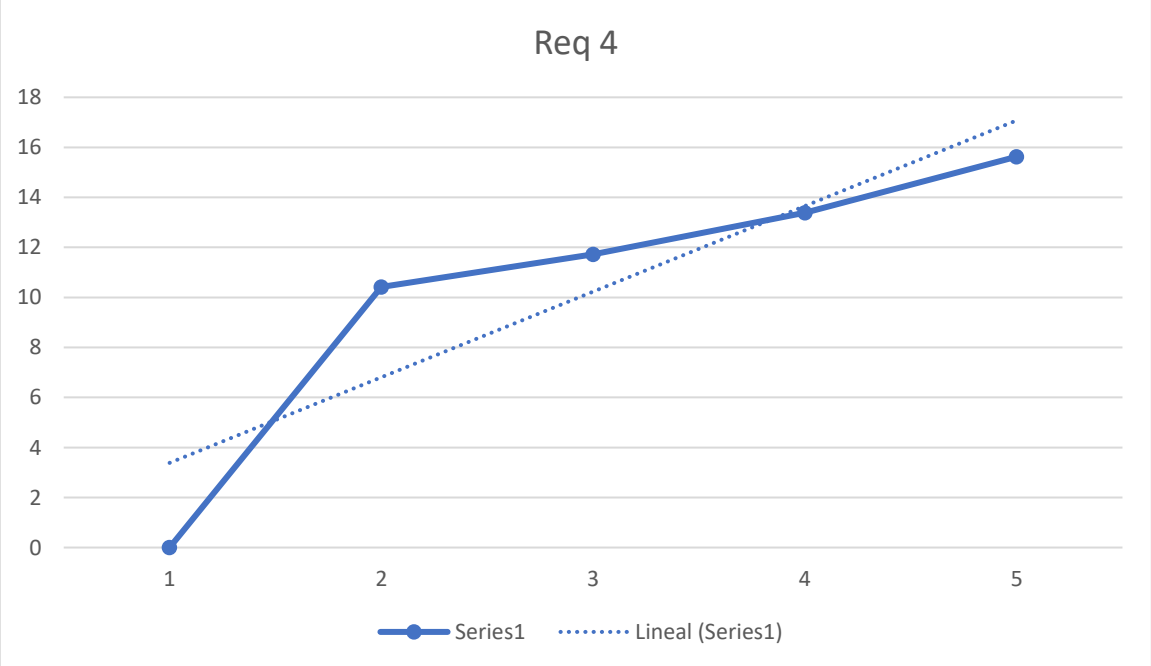
- La función contarAvistamientosDia tiene una complejidad temporal de O(m). Inicialmente se obtiene la primera llave del mapa ordenado por fechas para obtener la fecha con avistamientos más antigua, de donde se saca la cantidad de avistamientos de esa fecha con O (1). Posteriormente se usan la función dt.date, om.ceiling, om.floor y om.values para obtener los valores en el rango de fechas ingresado por el usuario con O(1), dentro del cual se recorre cada una y se va sumando la cantidad de avistamientos que tenga como valor cada llave fecha con lt.size con complejidad O(m). Finalmente extrae sus primeros y últimos tres avistamientos por fecha gracias a que es un mapa ordenado, en donde si llegan a haber varios avistamientos en una misma fecha, los ordena por hora con Merge Sort O (p log(p)), sin embargo, esta complejidad es muy pequeña al no haber muchos avistamientos por fecha en específico y al solo necesitarse extraer 3 avistamientos antes de romper el ciclo. Así que la complejidad temporal del cuarto requerimiento es:

  O(m)

- Pruebas de tiempo – Tablas

	Small	20 pct	50 pct	80 pct	Large
Req 4	0	10.416	11.72	13.39	15.625

- Gráficas



- Aunque en un inicio no parezca seguir ningún orden, realmente tiende a ser lineal, y si tiene ese comportamiento es debido a que, al demorarse tan poco tiempo, el módulo time toma esos valores como 0 en el archivo small, y tiende a aproximar mucho los valores a 0 o 15.625. En comparación a las otras funciones esta parece ser la más rápida, casi siendo O(1) debido a que aunque existan más fechas que horas específicas a diferencia del requerimiento 3, las funciones de ordenamiento en caso de empates toman mucho menos tiempo al no haber tantos avistamientos en una misma fecha.

Requerimiento 5

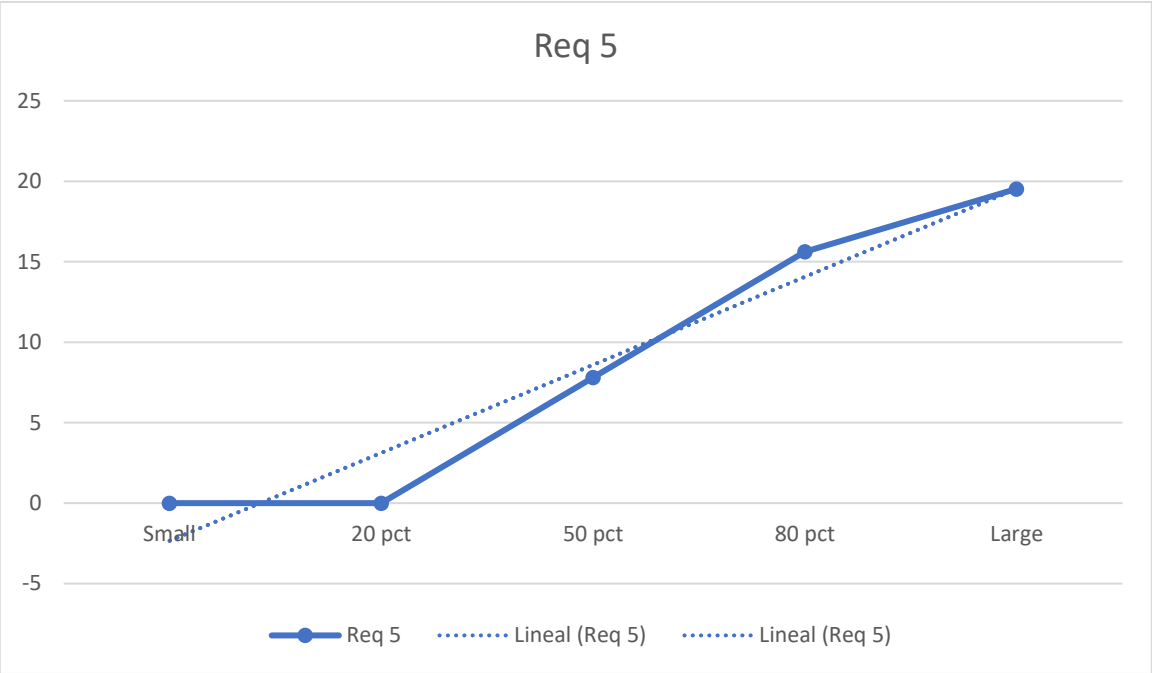
- Análisis complejidad temporal:
  M = Cantidad de llaves Longitud dentro del rango ingresado por el usuario
  P = Cantidad de llaves Latitud por llave Longitud dentro del rango ingresado por el usuario
- La función contarAvistamientosZona tiene una complejidad de O(M*P). Inicialmente se usan las funciones om.ceiling, om.floor y om.values para obtener los valores en el rango de longitudes ingresado por el usuario con O(1). Después, se realiza un recorrido de este resultado con O(M) y dentro de cada longitud del rango se extrae del mapa de latitudes asociado aquellas que estén dentro del rango ingresado por el usuario con O(1), usando nuevamente las funciones om.ceiling, om.floor y om.values. Posteriormente se recorren todas las latitudes asociadas a esa longitud con O(P) y se extraen los avistamientos asociados para añadirlos a la lista que se le retornara al usuario. Finalmente se obtiene el tamaño de la lista con lt.size y se extraen sus primeros y últimos cinco avistamientos ordenados por longitud y latitud gracias a que es un mapa ordenado. Así que la complejidad temporal del quinto requerimiento es:

  O(M*P)

- Pruebas de tiempo – Tablas

	Small	20 pct	50 pct	80 pct	Large
Req 5	0	0	7.812	15.625	19.53

- Gráficas



No se puede ver tan claramente su tendencia debido a que la función tarda muy poco en dar resultado debido a que M y P tienen valores pequeños y porque las acciones que se llevan a cabo para extraer el resultado son bastante simples.

Requerimiento 6

- Análisis complejidad temporal:

  La complejidad temporal es la misma del requerimiento 5 pues se llama a esta función, y el tiempo extra es debido al tiempo que consume la función folium.Map()

Aclaración Requerimiento 6

Después de correr el requerimiento 6 se podrá encontrar el archivo HTML del mapa en la carpeta Reto3-G02, donde posteriormente se podrá abrir en cualquier navegador. Este es un ejemplo de cómo se debería ver con los datos de entrada -103.00, -109.05, 31.33, 37.00: