ANÁLISIS DEL RETO

Jesús Ernesto Sandoval Santana, 202215917, Je.sandovals1@uniadnes.edu.co

Andrea Carolina Rodriguez Salinas, 202214026, ac.rodriguezs1@uniandes.edu.co

Hever André Alfonso Jiménez, 202220298, h.alfonsoj@uniandes.edu.co

Requerimiento 1

Descripción

Este requerimiento debe reportar todos los accidentes en un rango de fechas específico

Entrada	• Fecha inicial • fecha final
Salidas	• Número total de accidentes • una lista de todos los accidentes ocurridos
Implementado (Sí/No)	Si

Análisis de complejidad

Pasos	**Complejidad**
Buscar los elementos en un índice de la carga de datos con los valores de la fecha que necesitamos (om.values)	O (Log n + K)
Crear una nueva lista en donde van a ir los datos (lt.new _list)	O (1)
Recorrer la lista (lt.iterator)	O(n)
Recorrer cada elemento de la lista en su llave accidentes (lt.iterator)	O(n)
Verificar que esté presente (lt.isPresent)	O(n)
Añadir a la lista (lt.addLast)	O (1)
Organizar la lista (merge.sort)	O (n log n)
TOTAL	O (n log n)

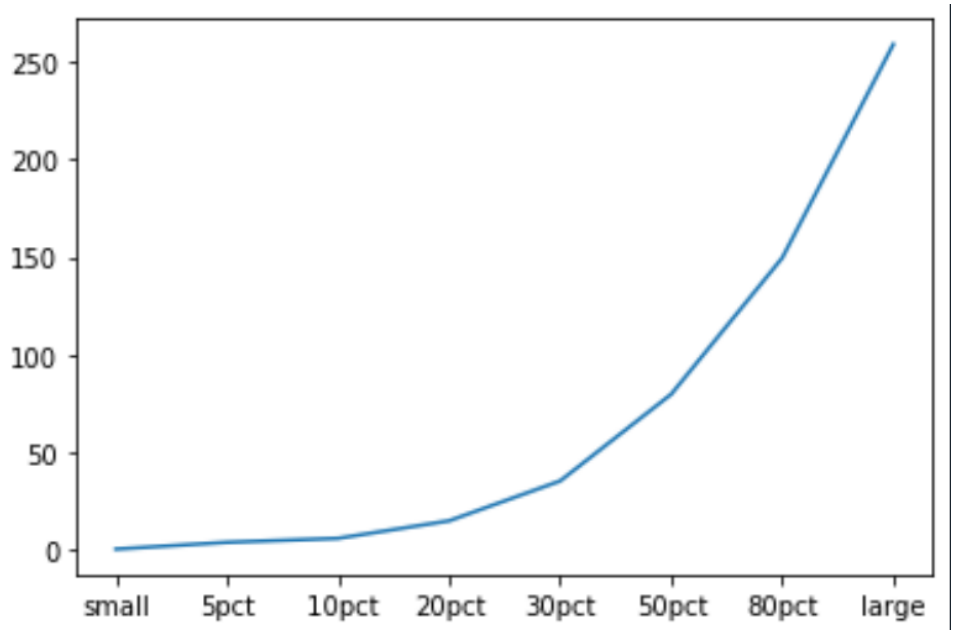
Pruebas Realizadas

Las pruebas de este requerimiento se hicieron con los siguientes datos: *Fecha inicial:* 01/11/2016, *Fecha final*: 08/11/2016.

Procesadores	Intel(R) Core (TM) i7-1165G7 2.80GHz
Memoria RAM	12.0 GB
Sistema Operativo	Windows 11

Entrada	**Tiempo (ms)**
small	0.638
5 pct	4.083
10 pct	6.115
20 pct	15.076
30 pct	35.460
50 pct	79.839
80 pct	149.623
large	258.825

Graficas



Análisis

Este algoritmo tiene como objetivo informar todos los eventos dentro de un rango de fechas específico. La complejidad de implementación del algoritmo (n log n) porque requiere recorrer la lista y realizar búsquedas y adiciones. Las pruebas realizadas muestran que el tiempo de ejecución aumenta con la cantidad de datos.

Requerimiento 2

Descripción

Este requerimiento debe reportar la cantidad de accidentes ocurridos en un intervalo de tiempo de un día específico

Entrada	• Horas y minutos iniciales del intervalo • hora y minutos finales del intervalo • año de consulta y mes de consulta
Salidas	• El número total de accidentes ocurridos en el intervalo de tiempo, año y mes dados. • lista de todos los accidentes ordenados del menos reciente al más reciente por su hora y minutos de ocurrencia
Implementado (Sí/No)	Si

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Traer los datos del data_struct del año deseado(om.get)	O (log n)
Sacar el índice del mes del árbol creado en el paso anterior(me.getValue)	O (1)
Traer los datos del mes que nos interesa del árbol creado en el paso anterior(om.get)	O (log n)
sacar el índice de las horas del árbol creado en el paso anterior(me.getValue)	O (1)
Traer la hora que nos interesa de ese árbol (om.values)	O (log n + k)
Se crea una lista vacía para almacenar los datos que deseamos mostrar(lt.newList)	O (1)
Se itera el arbol con las horas deseadas (lt.itertator)	O(n)
Se itera cada línea del paso anterior (lt.iterator)	O(n)
Se verifica si este está presente dentro de la lista(lt.ispresent)	O (n)
Si el “if” anterior es diferente de 0 se añade a la lista anterior(addLast)	O (1)
Se hace merge del resultado según las fechas de los accidentes(merge.sort)	O (n log n)
TOTAL	O (n log n)

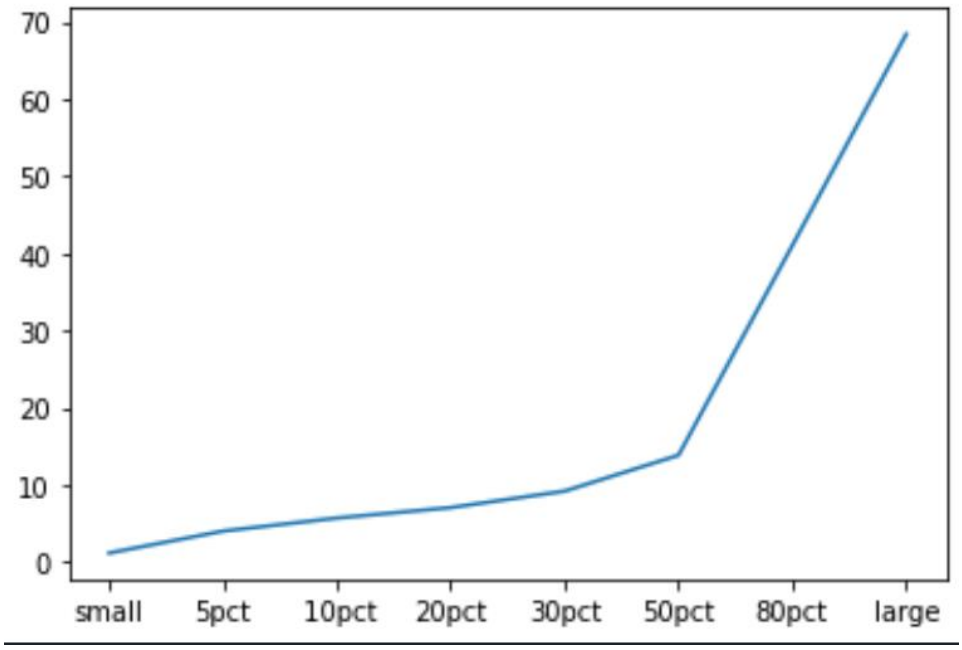
Pruebas Realizadas

Las pruebas fueron realizadas con los datos: *mes: 11, año :2022, hora inicial: 18:00, hora final: 21:00*

Procesadores	Intel(R) Core (TM) i7-1165G7 2.80GHz
Memoria RAM	12.0 GB
Sistema Operativo	Windows 11

Entrada	**Tiempo (ms)**
small	1.167
5 pct	4.007
10 pct	5.700
20 pct	7.050
30 pct	9.189
50 pct	13.836
80 pct	41.017
large	68.399

Graficas



Análisis

Este algoritmo está diseñado para informar la cantidad de eventos que ocurrieron en un intervalo de tiempo en una fecha determinada y proporcionar una lista de todos los eventos ordenados por horas y minutos. El análisis muestra que la complejidad del algoritmo es O (n log n). Las pruebas en una computadora con un procesador Intel(R) Core (TM) i7-1165G7 de 2,80 GHz y 12 GB de RAM mostraron tiempos de respuesta adecuados para varios tamaños de entrada.

Requerimiento 3(Individual)

Descripción

En este requerimiento se quiere saber los tres accidentes más recientes de una clase especifica en una vía de la ciudad

Entrada	• Clase del accidente • Nombre de la vía
Salidas	• El número total de accidentes que pasaron en esta vía • Los tres accidentes más recientes
Implementado (Sí/No)	Si (HEVER ANDRE ALFONSO)

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Se crea una lista vacía (lt.newList)	O (1)
Se itera la lista en su llave accidentes(lt.iterator)	O(n)
Se compara cada línea con base a la clase de accidente dado (if)	O (1)
Corta los elementos iterados en el punto anterior en su llave dirección (. Split)	O (1)
Comprar si la vía está dentro de la cadena de caracteres anterior (in)	O (1)
Si este es verdad lo añade de ultimo a la lista original (lt.addLast)	O (1)
Ordenamos la lista (merg.sort)	O (n log n)
Se hace una sub lista con los 3 primeros para mostrar (subList)	O (1)
Total	O(n log n)

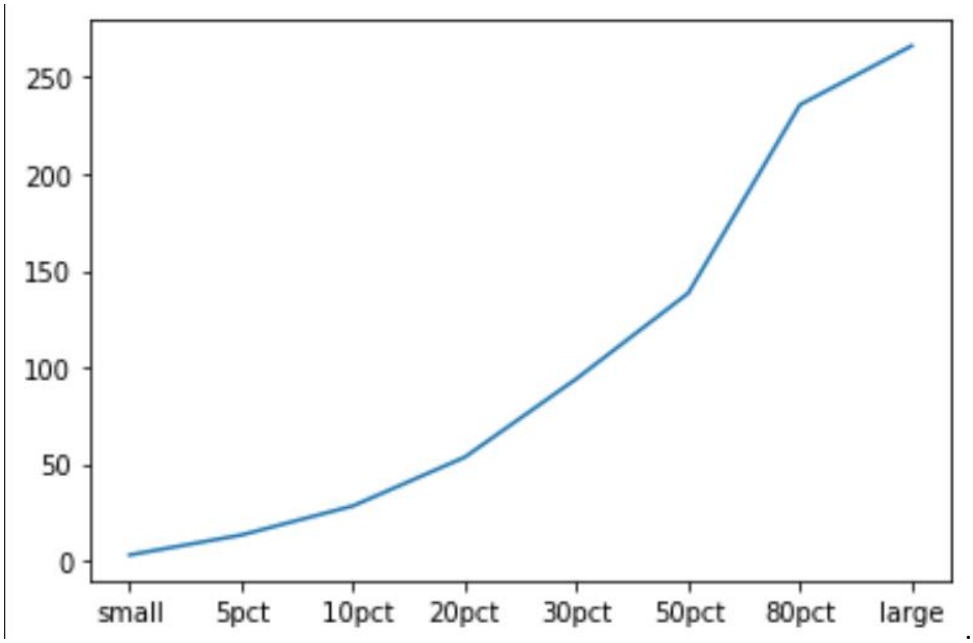
Pruebas Realizadas

Las pruebas fueron realizadas con los datos: *clase: choque, Vía: Avenida las Américas*

Procesadores	Intel(R) Core (TM) i7-1165G7 2.80GHz
Memoria RAM	12.0 GB
Sistema Operativo	Windows 11

Entrada	**Tiempo (ms)**
small	2.997
5 pct	13.307
10 pct	28.356
20 pct	53.556
30 pct	94.129
50 pct	138.352
80 pct	235.783
large	266.271

Graficas



Análisis

Este algoritmo busca los tres accidentes más recientes en una categoría determinada en las vías de la ciudad. Primero, recorra la lista de eventos y verifique que cada evento pertenezca a una clase específica y tenga una ruta específica. Si es así, se añadirá a una lista vacía. Luego, la lista se ordena por fecha y se crea una sublista que muestra los tres incidentes más recientes. La complejidad del algoritmo es O(n log n), donde n es el número de ocurrencias en la lista. Las pruebas se ejecutan con diferentes tamaños de entrada y el tiempo de ejecución está dentro de límites razonables.

Requerimiento 4 (INDIVIDUAL)

Descripción

Se quiere saber la cantidad dad de accidentes y los 5 accidentes más recientes en un intervalo de fechas con base a su gravedad

Entrada	• Fecha de inicio del intervalo • Fecha del final de intervalo • Gravedad del accidente
Salidas	• Número de accidentes ocurridos • Lista con los 5 accidentes más recientes
Implementado (Sí/No)	Si (ANDREA CAROLINA RODRIGUEZ)

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Traer los valores del data_struct en el rango deseado según la fecha dada	O (log n + k)
Se crea una lista nueva en la cual se van a guardar los datos	O(1)
Se itera el mapa anterior	O(n)
Se trae el índice de gravedad que coincida con el dado por el usuario	O (n)
Se hace un “if” en el cual si el dato anterior es diferente de “None” entra en la condición	O (1)
Se saca el valor de mapa	O(1)
Se saca la información del mapa necesaria del mapa anterior	O (1)
Y se añade de primero en la lista	O (1)
Se saca el tamaño	O (1)
Se hace otro if en el cual se compara si el tamaño de la lista es mayor a 5 se hace una “sublist” de los primeros 5	O (1)
Y si la condición anterior no es cierta solamente retorna la lista original	O (1)
TOTAL	***O (n^2)***

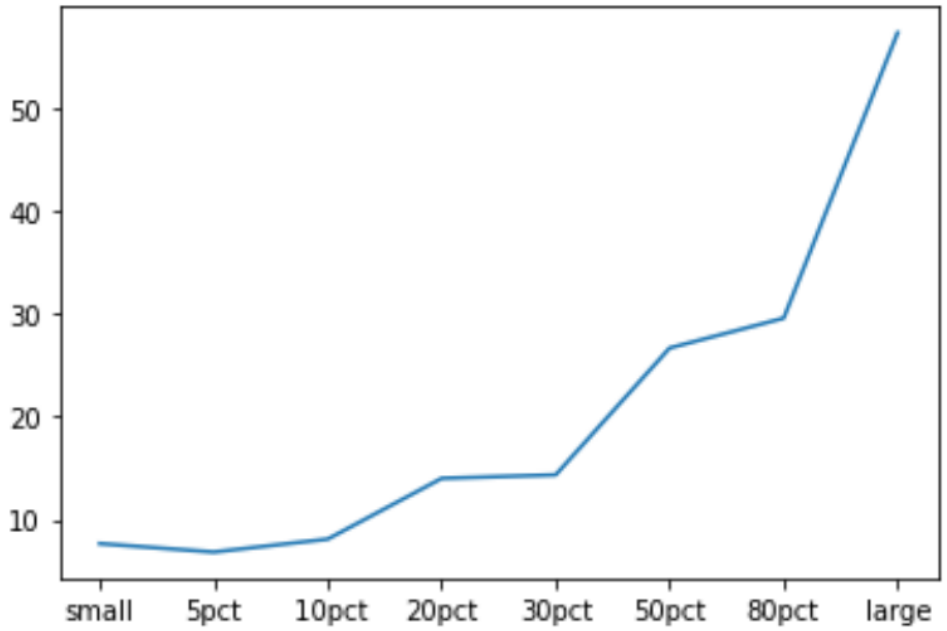
Pruebas Realizadas

Las pruebas fueron realizadas con los datos: *fecha inicia: 01/10/2016, fecha final: 01/10/2018, gravedad: "CON MUERTOS"*

Procesadores	Intel(R) Core (TM) i7-1165G7 2.80GHz
Memoria RAM	12.0 GB
Sistema Operativo	Windows 11

Entrada	**Tiempo (ms)**
small	7.675
5 pct	6.856
10 pct	8.109
20 pct	14.018
30 pct	14.356
50 pct	26.693
80 pct	29.587
large	57.343

Graficas



Análisis

Este algoritmo encuentra el evento más reciente con una gravedad dada dentro de un rango de fechas y devuelve una lista de hasta 5 eventos. Seleccione un valor de intervalo y busque en la estructura de datos eventos de la gravedad deseada. La complejidad es O (n***^2***), donde n es el tamaño de los datos del mapa. Las pruebas muestran un buen rendimiento de funcionamiento.

Requerimiento 5(INDIVIDUAL)

Descripción

Reporta los 10 accidentes más recientes en un mes, un año y una localidad específicos

Entrada	• La localidad deseada • Mes • Un año entre 2015 y 2022
Salidas	• Número total de accidentes ocurridos en la localidad durante la fecha indicada
Implementado (Sí/No)	Si (JESUS SANDOVAL)

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Traer los datos del data_struct del año deseado (om.get)	O (log n)
Sacar el índice del mes del árbol creado en el paso anterior(me.getValue)	O (1)
Traer los datos del mes que nos interesa del árbol creado en el paso anterior(om.get)	O (log n)
sacar el índice de la localidad del árbol creado en el paso anterior(me.getValue)	O (1)
Se trae la localidad deseada del arbol anterior(mp.get)	O (log n)
Se saca el valor del paso anterior(me.getValue)	O (1)
Hacemos “merge sort” de la localidad deseada(merge.sort)	O (n log n)
Si es menor a 10 se retorna la lista completa(lt.size)	O (1)
Si no se hace una sub lista de tamaño 10 (lt.sublist)	O (1)
TOTAL	***O (n log n)***

Pruebas Realizadas

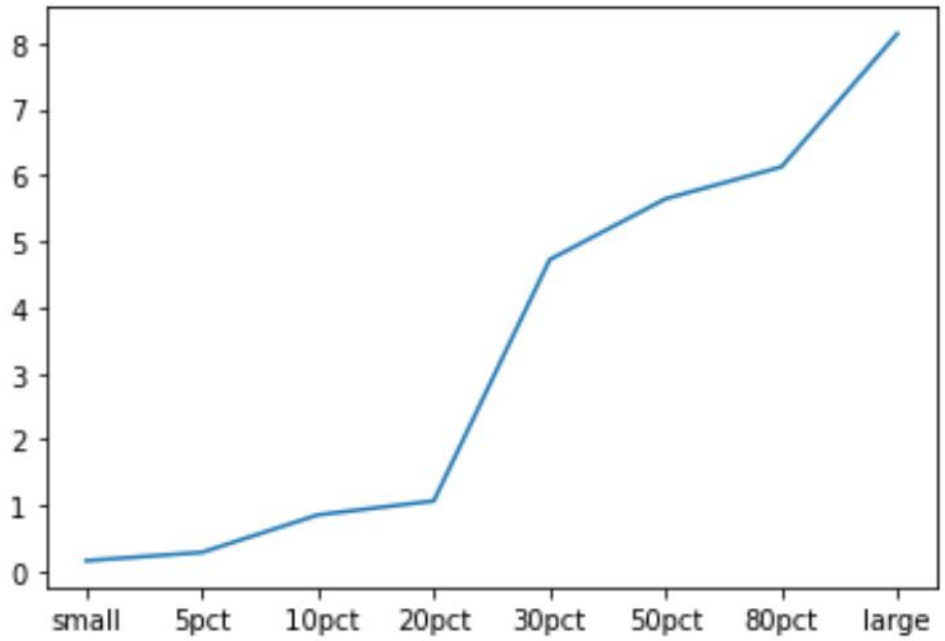
Las pruebas fueron realizadas con los datos: *mes: 11, localidad: Fontibón, año: 2016,*

Procesadores	Intel(R) Core (TM) i7-1165G7 2.80GHz
Memoria RAM	12.0 GB
Sistema Operativo	Windows 11

Entrada	**Tiempo (ms)**

small	0.165
5 pct	0.291
10 pct	0.859
20 pct	1.070
30 pct	4.724
50 pct	5.645
80 pct	6.128
large	8.146

Graficas



Análisis

Este algoritmo busca los últimos 10 accidentes en un lugar, mes y año determinados. Primero accede a los datos del año requerido, luego busca datos mensuales y datos locales, y finalmente ordena y selecciona los 10 más cercanos. El algoritmo tiene un buen desempeño en términos de tiempo de ejecución con una complejidad de O (n log n), donde n es el tamaño de los datos a procesar.

Requerimiento 6

Descripción

Este requerimiento debe mostrar la cantidad de accidentes dados por el usuario dadas en una zona circula especifica de la ciudad para un mes y un año específicos

Entrada	• Mes • Año • Coordenadas (latitud y longitud) • Radio del área • Número de accidentes que desea mostrar
Salidas	• Los n accidentes organizados por el más cercano al centro de la zona dada • Una lista de todos los accidentes
Implementado (Sí/No)	Si

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Llamamos la función global para cambiar la función de comparación que vamos a usar en el merg más adelante(global)	O (1)
Llamamos la función global para cambiar la función de comparación que vamos a usar en el merg más adelante (global)	O (1)
asignación	O (1)
asignación	O (1)
Trae el índice que necesitamos con base al año deseado (om.get)	O (log n)
Saca el valor del índice en su subíndice meses(me.getValue)	O (1)
Trae el índice con base al mes deseado(om.get)	O (log n)
Trae el valor del mapa anterior en su subindice lista de accidente (me.getValue)	O (1)
Se ordena la lista con base a la distancia (merge sort)	O (n log n)
Se hace una sublista con el número de accidentes que desea el usuario (sublist)	O(n)
Se hace una lista nueva (lt.newlist)	O (1)
Se itera la lista (lt.iterator)	O(n)

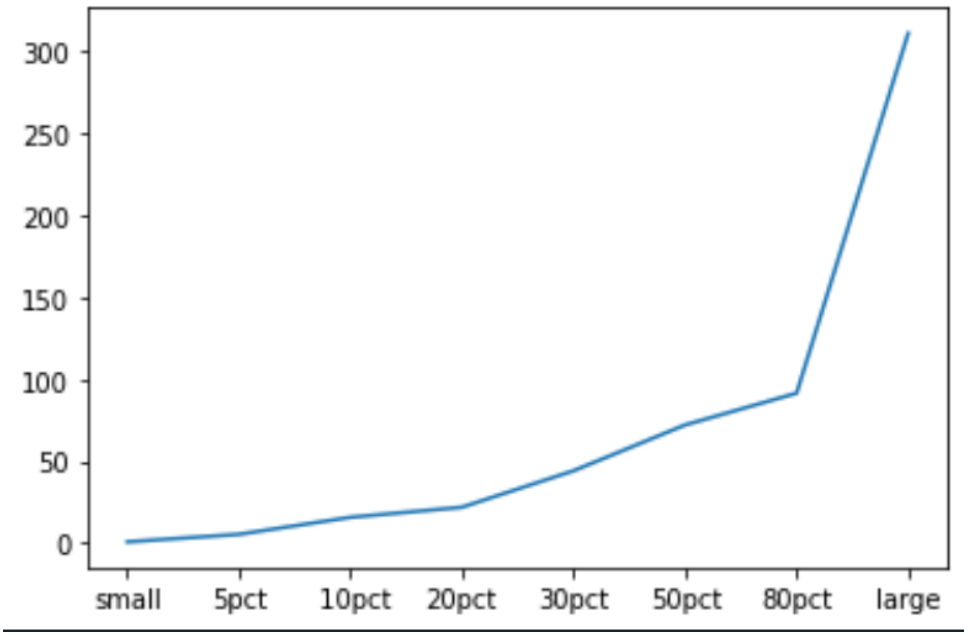
Se compara la distancia de cada elemento con base al centro del radio(if)	O (1)
Esto hace que cuando el radio sea mayor lo deje de ciclar debido a que después de este dato todas las distancias van a ser mayores que el radio esperado (else y break)	O (1)
Si el tamaño de la lista es menor a la cantidad esperada imprime la lista original (If)	O (1)
Si es mayor retorna False (else)	O (1)
TOTAL	***O(n log n)***

Pruebas Realizadas

Las pruebas fueron realizadas los datos: *número deseado por el usuario("Top"): 3, latitud: 4.674, longitud: -74,068, radio: 5 km, mes: 01, año: 2022.*

Procesadores	Intel(R) Core (TM) i7-1165G7 2.80GHz
Memoria RAM	12.0 GB
Sistema Operativo	Windows 11

Entrada	**Tiempo (ms)**
small	0.908
5 pct	5.479
10 pct	15.866
20 pct	21.995
30 pct	44.224
50 pct	72.165
80 pct	91.532
large	310.899



Graficas

Las gráficas con la representación de las pruebas realizadas.

Análisis

Este algoritmo busca mostrar la cantidad de accidentes en un área circular específica de la ciudad para un mes y año específicos. La complejidad algorítmica para esta tarea es O (n log n). Ha sido probado con varios tamaños de entrada, de pequeño a grande, el tiempo de procesamiento aumenta significativamente a medida que aumenta el tamaño de entrada. Las pruebas se realizaron en Windows 11 con un procesador Intel Core i7-1165G7 y 12 GB de RAM.

Requerimiento 7

Descripción

En este requerimiento debe mostrar los accidentes más temprano y más tarde para cada día de un mes y año específico.

Entrada	• Mes • Año • Coordenadas (latitud y longitud) • Radio del área • Número de accidentes que desea mostrar
Salidas	• Los n accidentes organizados por el más cercano al centro de la zona dada • Una lista de todos los accidentes
Implementado (Sí/No)	Si

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Trae el indice según el año deseado (om.get)	O(1)
Saca el valor del arbol anterior(me.getValue)	O(log n)
Trae el indice del mes deseado (om.get)	O(1)
Saca el valor del arbol anterior (me.getValue)	O(log n)
Trae las llaves del mapa anterior(om.keyset)	O(n)
Itera la lista de llaves (lt.iterator)	O(n)
Crea una nueva lista dentro del ciclo(lt.newlist)	O(1)
Según el día trae los datos del arbol de los días (om.get)	O(1)
Lo filtra por su indice de hora (me.getValue)	O(log n)
Se retorna la llave menor de la tabla de hash (om.minkey)	O(1)

Trae el indice de la llave menor (om.get)	O(1)
Trae el valor de la llave menor(me.getValue)	O(1)
Se retorna la llave mayor de la tabla de hash (om.maxkey)	O(1)
Trae el indice de la llave mayor (om.get)	O(1)
Trae el valor de la llave mayor (me.getValue)	O(1)
Se hace merge con los primeros 3(merge.sort)	O(n log n)
Se hace merge con los últimos 3(merge.sort)	O(n log n)
Se saca el primero de la lista de los primeros(firstelement)	O(1)
Se saca el primero de la lista de los ultimos(firstelement)	O(1)
Se añade de ultimo a la lista creada al inicio dentro del for el primer elemento de los primeros(addlast)	O(1)
Se añade de ultimo a la lista creada al inicio dentro del for el primer elemento de los ultimos(addlast)	O(1)
Se añade de ultimo esta lista a la lista antes del for(addlast)	O(1)
Se hace una lista de franjas horarias	O(1)
Del mapa del mes se saca el valor de las horas(getValue)	O(1)
Se crea una lista nueva	O(1)
Se itera la lista de las franjas horarias (for)	O(n)
Se saca la hora inicial	O(1)
Se saca la hora final	O(1)
Saca los valores en un rango dado (om.values)	O(n)
Se crea un contador	O(1)
Se itera la lista de las horas en un rango (lt.iterator)	O(n)
El contador actualiza dependiendo del tamaño de la hora	O(n)
Agrega el tamaño de la lista a la lista creada al inicio del for (lt.append)	O(1)
TOTAL	***O(n log n)***

Pruebas Realizadas

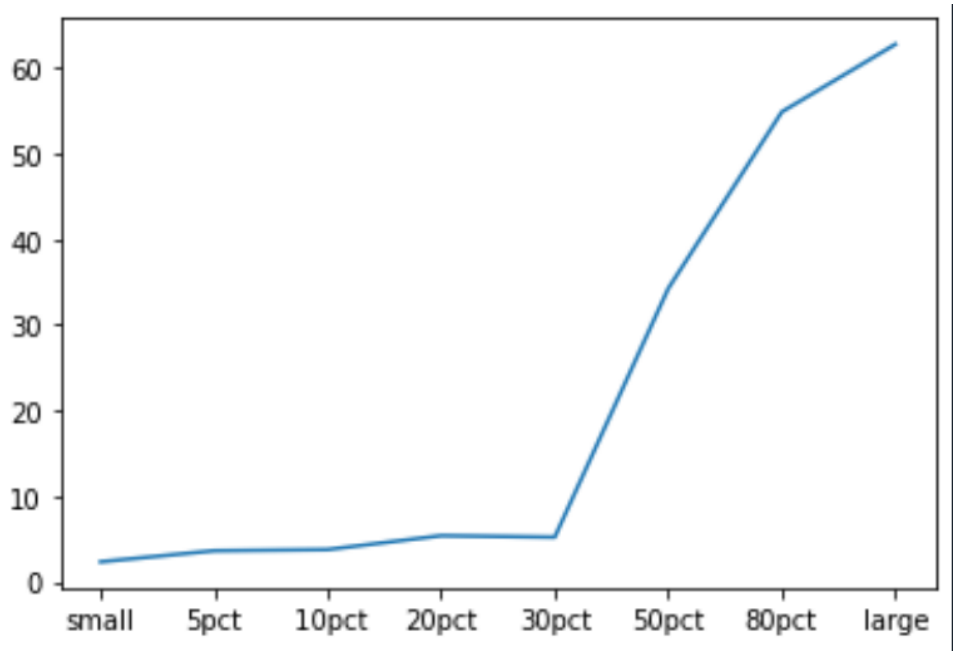
Las pruebas fueron realizadas con los datos: *mes: 12, año: 2019*

Procesadores	Intel(R) Core (TM) i7-1165G7 2.80GHz
Memoria RAM	12.0 GB
Sistema Operativo	Windows 11

Entrada	**Tiempo (ms)**
small	2.345
5 pct	3.632

10 pct	3.771
20 pct	5.396
30 pct	5.224
50 pct	34.260
80 pct	54.886
large	62.781

Graficas



Análisis

Este algoritmo muestra los accidentes más antiguos y más recientes para cada día del mes y del año en un área circular determinada. La complejidad del algoritmo es O(n log n) y ha sido probado con diferentes tamaños de entrada, el tiempo de procesamiento aumenta significativamente a medida que aumenta el tamaño de entrada. Las pruebas se realizaron en una computadora con un procesador Intel Core i7-1165G7 y 12 GB de RAM con Windows 11.

Requerimiento 8 (BONO)

Descripción

Este requerimiento tiene como finalidad visualizar los accidentes de cierta clase ocurridos en un rango de fechas específicas

Entrada	• Fecha inicial • Fecha final
Salidas	• Número total de accidentes • Mapa interactivo que muestre todos los accidentes según su tipo en el mapa de Bogotá
Implementado (Sí/No)	Si

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Saca un rango de valores de la lista en el indice deseado (om.values)	O(log n)
Se crea una nueva lista(newlist)	O(1)
Se crea un contador de accidentes	O(1)
Se itera la lista(lt.newlist)	O(n)
Se le suma el tamaño de la lista accidentes al contador anterior	O(1)
Se llama la lista en el indice deseado (lista["classIndex"])	O(1)
Se trae la clase necesaria de la lista anterior(mp.get)	O(1)
Se trae el valor de la lista anterior (me.getvalue)	O(1)
Se itera la lista (lt.iterator)	O(n)
Se verifica si el elemento está dentro de la lista inicial (lt.ispresent)	O(n)
Si no está en la lista inicial la añade(addlast)	O(n)
Se crea el mapa para mostrar en la consola (folium. Map)	O(1)
Se itera en la lista filtrada en el for anterior(lt.iterator)	O(n)
Se crea un diccionario con las gravedades con cierto color	O(1)
Se crea el icono para cada objeto de la lista dentro del iterador según la gravedad de cada elemento (folium.Icon)	O(n)
Se crea un marcador dentro del mapa (folium.Marker)	O(n)
Se añade un marcador al mapa(add_to)	O(n)
TOTAL	***O(n^2)***

Pruebas Realizadas

Fecha inicial 28/01/2019 , Fecha final 17/07/2021, Clase CHOQUE

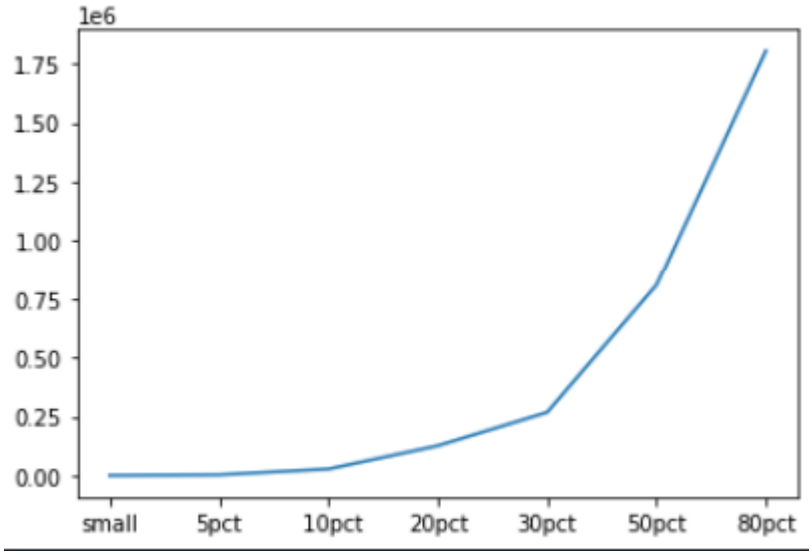
Procesadores	Intel(R) Core (TM) i7-1165G7 2.80GHz
Memoria RAM	12.0 GB
Sistema Operativo	Windows 11

Entrada	**Tiempo (ms)**
small	190.061
5 pct	2744.035

10 pct	27185.142
20 pct	125655.692
30 pct	266887.074
50 pct	806576.982
80 pct	1803503.657
large	No aplica

En el inciso de large no pusimos dato debido a que este demoraba mucho mas de 30 minutos compilando en la maquina usada para todas las pruebas del reto

Graficas



Análisis

Este algoritmo es O (n) ^2 esto genera que al ponerlo en funcionamiento tenga una duración de tiempos cada vez mayor, a diferencia de los vistos anteriormente. Esto debido a que se tiene que recorrer la lista aproximadamente 2 veces para que se pueda llevar a cabo el proceso de graficarlo en el mapa, añadido a esto se puede observar que comparado con la gráfica se puede ver un espectro muy similar a la complejidad esperada inicialmente. Este se hace con los datos Fecha inicial 28/01/2019, Fecha final 17/07/2021, Clase CHOQUE.