

Departamento de Ingeniería de Sistemas y Computación Estructuras de Datos y Algoritmos ISIS-1225



# **ANÁLISIS DEL RETO**

Harold Esteban Piñeros Monroy, 202316402, h.pineros@uniandes.edu.co
Carlos Alberto Poveda Riaño, 202315546, ca.povedar1@uniandes.edu.co
Luis Sebastián Contreras Diaz, 202311819, ls.contreras@uniandes.edu.co

# **Requerimiento 1**

# Descripción

```
getDatesByRange(analyzer, initialDate, finalDate):
Retorna el numero de crimenes en un rago de fechas.
final = lt.newList('SINGLE_LINKED')
initialDate = datetime.datetime.strptime(initialDate, '%Y-%m-%dT%H:%M')
finalDate = datetime.datetime.strptime(finalDate, '%Y-%m-%dT%H:%M')
lst = om.values(analyzer, initialDate, finalDate)
totearthquakes = lt.size(lst)
events = 0
for lstdate in lt.iterator(lst):
    for j in lt.iterator(lstdate):
        time = j['time']
        events += 1
        dic[time] = {
            'time':time,
            'events':1,
            'details':j
        lt.addFirst(final,dic[time])
return totearthquakes, final, events
```

Este requerimiento se encarga de ver los eventos sísmicos mundiales ocurridos durante un intervalo de fechas específico.

2	
Entrada	Fecha inicial del intervalo (en formato "%Y-%m-%dT%H:%M"). • Fecha final del intervalo (en formato "%Y-%m-%dT%H:%M"). • La
	significancia mínima del evento (sig).
Salidas	El número total de eventos sísmicos ocurridos durante las fechas indicadas. • Todos los eventos ocurridos en el intervalo ordenados cronológicamente desde el más reciente al más antiguo.
Implementado (Sí/No)	Si. Implementado por Luis Sebastián Contreras

# Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	Complejidad
Inicialización de estructuras de datos y conversiones de fechas:	O(1)
Obtención de elementos según rango de fechas:	O(log(M))
Ciclo anidado for para recorrer la lista de valores	O(M*M)
Creación y adición de elementos a una lista	O(1)
TOTAL	O(M*M)

# **Pruebas Realizadas**

Las pruebas realizadas fueron realizadas en una maquina con las siguientes especificaciones. Los datos de entrada fueron fecha inicial 1999-03-21T05:00 y fecha final 2004-10-23T17:30.

Procesadores	12th Gen Intel(R) Core(TM) i5-1235U
Memoria RAM	16 GB
Sistema Operativo	Windows 11 Home

Entrada	Tiempo (ms)
small	12.81
5 pct	145.29
5 pct 10 pct	207.78
20 pct	254.37
30 pct	805.64
50 pct	1454.28
80 pct	4837.21
large	12750.25

#### **Tablas de datos**

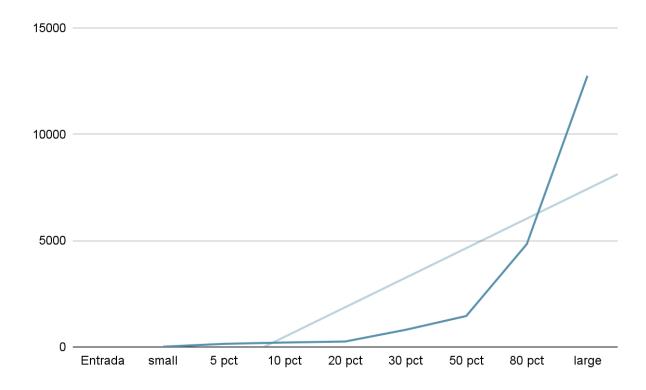
		Tiempo
Muestra	Salida	

	otal de eventos: 912	
small	time   events	12.81
5 pct	time	145.29
10 pct	1   mag   long   lat   depth   sig   nst   title   cdi   mm1   magType   type   code	207.78

		1
20 pct	Table   Central   Centra	254.37
30 pct	Time	805.64
50 pct	time   events	1454.28

	total de eventos: /4/91	
	time   events   details	
80 pct	2004-10-23717:27:08.2000002	
	2894-18-23716:35:43.59090827   1	
	2004-10-23716:32:27.7100002	4837.21
	1999-63-21709:16:19.9609082	4037.21
	1999-83-21109:02:48.19808007   1	
	1999-83-21705:15:14.6980807   1	
	Bi enventido	
large	time   events	12750.25

Las gráficas con la representación de las pruebas realizadas.



### **Análisis**

A pesar de que obtener un elemento en un *arbo*l tiene una complejidad lineal O(N), la implementación incurre en ciclos anidados, lo cual lo puede hacer complejo,  $O(M^2)$ .

Este comportamiento se puede evidenciar experimentalmente en la gráfica. Ya que, gracias a que los datos no se encuentran tan dispersos con respecto a la línea de tendencia, la curva coincide con el comportamiento exponencial esperado.

# Requerimiento 2

```
def req_2(analyzer, initialmag, finalmag):
   Retorna el numero de crimenes en un rago de fechas.
   final = lt.newList('ARRAY LIST')
   dic = \{\}
   lst = om.values(analyzer, initialmag, finalmag)
   totearthquakes = lt.size(lst)
   events = 0
    for lstdate in lt.iterator(lst):
        for j in lt.iterator(lstdate):
            mag = j['mag']
            events += 1
            dic[mag] = {
                'mag':mag,
                'events':1,
                'details':j
            lt.addFirst(final,dic[mag])
   return totearthquakes, final, events
```

# Descripción

Este requerimiento se encarga de ver los eventos sísmicos mundiales ocurridos durante un intervalo de fechas específico.

Entrada	Magnitud inicial del intervalo(mag) • Magnitud final del intervalo(mag)
Salidas	El número total de eventos sísmicos ocurridos dentro del intervalo indicado. • Todos los eventos ocurridos en el intervalo ordenados cronológicamente desde el más reciente al más antiguo.
Implementado (Sí/No)	Si. Implementado por Harold Piñeros Monroy

# Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	Complejidad
Inicialización de estructuras de datos y conversiones de fechas:	O(1)
Obtención de elementos según rango demagnitud:	O(log(M))
Ciclo anidado for para recorrer la lista de valores	O(M*M)
Creación y adición de elementos a una lista	O(1)
TOTAL	O(M*M)

# **Pruebas Realizadas**

Las pruebas realizadas fueron realizadas en una maquina con las siguientes especificaciones. Los datos de entrada fueron magnitud inicial 3.5 y magnitud final 6.5.

Procesadores	8th Gen Intel(R) Core(TM) i7-8565U
Memoria RAM	16 GB
Sistema Operativo	Windows 11 Home

Entrada	Tiempo (ms)
small	79.15
5 pct	298.87
10 pct	1571.45
20 pct	4504.60
30 pct	6616.65
50 pct	14659.79
80 pct	49492.91
large	61633.50

#### **Tablas de datos**

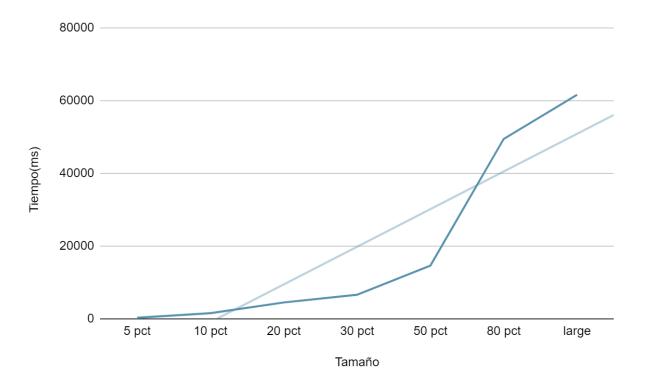
		Tiempo
Muestra	Salida	

small		79.15
5 pct	mag	298.87
10 pct	Mag   events	1571.45

		1
20 pct		4504.60
	time   long   lat   depth   sig   nst   title   cdi   mmi   magfype   type   code	
	2023-65-22781:48:27.5680802   -104.29757   31.687548   6.6744554   180   20.0   M 3.5 - 54 km 5 of Whites City, New Mexico   2.6   4.668   ml   carthquake   2023jstg	
	2016-02-05718:28-42.7690002   -145-8129   59.5548   25.4   188   M 3.5 - 118 tem 5 of Condonny, Alaska     el.   earthquake   0161nufedek	
	mag   events	
	6.5   1	
30 pct	6.5 1   time   long   lat   depth   sig   nst   title   cdi   msi   msg/yee   type   code	6616.65
	time long lat depth sig nst title [cdi mmi magfype type code ]   2885-11-21783:25:47.0690002   -164.044   53.849   51.8   188   26.0   M 3.5 - 117 km ESE of Abutam, Alaska   ml earthquake   p8000e4md	
	time   long   lat   depth   sig   not   title   cdi   med   englype   type   code	
	1.5   1   time   long   lat   depth   sig   mst   title   cdi   mst   mag/lype   type   code	
	mag   events	
	6.5 1   time   long   lat   depth   sig   nst   title   cdi   mmi   magType   type   code     2003-03-25702:53:25.0300002   120.743   -8.204   33.0   650   247.0   M 6.5 - 46 km NC of Ruteng, Indonesia   6.584   mac   earthquake   p800cht5	
50 pct	6.5 1 time long lat depth sig not title citi mms magType type code   2005-02-19700-04:43.5900002 122.129 5.562 10.0 650 371.0 M 6.5 - 81 km SSM of Katabu, Indonesia 6.828 mub carthquake p8000gfv	
	6.5 1   time   long   lat   depth sig   nst   title   cdi   mmi   mmg/lyne   type   code	14659.79
	3.5 1   time   long   lat   depth   sig   nst   title   cdi   mmi   mag/lype   type   code	
	3.5 1   time   long   lat   depth   sig   nst   title   cdi   mmi   magType   type   code	
	1.5   1   time   long   lat   depth   sig   nst   title   cdi   msi   msglyne   type   code	

80 pct	Mag   Seconds	49492.91
large	mag   events	61633.50

Las gráficas con la representación de las pruebas realizadas.



# **Análisis**

A pesar de que obtener un elemento en un *arbo*l tiene una complejidad lineal O(N), la implementación incurre en ciclos anidados, lo cual lo puede hacer complejo, O(Y^2).

Este comportamiento se puede evidenciar experimentalmente en la gráfica. Ya que, gracias a que los datos no se encuentran tan dispersos con respecto a la línea de tendencia, la curva coincide con el comportamiento exponencial esperado.

# **Requerimiento 3**

```
def req_3(min_mag,max_depth,analyzer):
    Función que soluciona el requerimiento 3
    final = lt.newList('ARRAY_LIST')
   newLista = lt.newList('ARRAY_LIST')
   hp = heap.newHeap(compare_dicts)
   dic = {}
   data_structs = analyzer['mag']
   x = om.values(data_structs,float(min_mag), float(om.maxKey(data_structs)))
    for i in lt.iterator(x):
        f = om.values(i,float(om.minKey(i)),float(max_depth))
        for j in lt.iterator(f):
            for z in lt.iterator(j):
               if len(z['depth'])>0:
                 if float(z['depth'])>0:
                    lt.addFirst(newLista,z)
    sa.sort(newLista,compareReq3)
    a = lt.subList(newLista,1,17)
    for z in lt.iterator(a):
       time = z['time']
       dic[time] = {
            'time':time,
            'events':1,
            'details':z
        lt.addLast(final,dic[time])
    return final
```

# Descripción

Este requerimiento se encarga de o consultar los 15 eventos sísmicos más recientes ocurridos que superen una significancia mínima y que sean menores a una distancia azimutal indicada.

Entrada	• La magnitud mínima del evento (mag). • La profundidad máxima del evento (depth).
Salidas	El número total de eventos sísmicos registrados mayores a la magnitud y menores a la profundidad indicada • Los diez (10)

	eventos cronológicamente más recientes que cumplan con los parámetros especificados. Cada uno de los eventos en la consulta
Implementado (Sí/No)	Si. Implementado por Harold Piñeros Monroy

# Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	Complejidad
Creación de listas y diccionarios (tiempo constante):	O(1)
Esto incluye inicializar las estructuras de datos como	
listas y diccionarios.	
Buscar valores en un árbol binario	O(log(N))
Ciclo anidado for para recorrer la lista de valores	O(M)
Buscar valores en un árbol binario	O(log(M))
Ciclo anidado for para recorrer los valores	O(Y)
Verificación de condiciones y operaciones	O(1)
condicionales (tiempo constante)	
Ciclo anidado for para recorrer los valores	O(Y*Y)
Añadir al final de la lista los elementos	O(1)
Ordenamiento de la lista (sa.sort)	O(Y(log(Y))
TOTAL	O(Y*(Y))

#### **Pruebas Realizadas**

Las pruebas realizadas fueron realizadas en una maquina con las siguientes especificaciones. Los datos de entrada fueron significancia mínima 300.0 y distancia azímutal máxima de 48.0.

Procesadores	8th Gen Intel(R) Core(TM) i7-8565U
Memoria RAM	16 GB
Sistema Operativo	Windows 11 Home

Entrada	Tiempo (ms)
small	8.81
5 pct	54.41
10 pct	130.05
20 pct	295.83
30 pct	512.05
50 pct	2416.42
80 pct	3438.75
large	6724.65

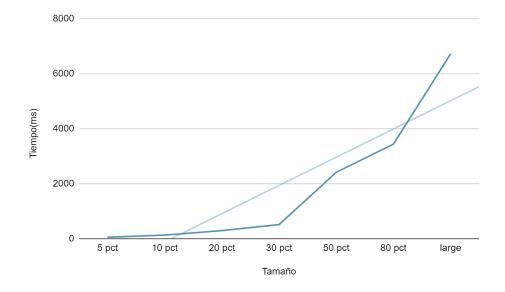
#### **Tablas de datos**

		Tiempo
Muestra	Salida	
	time   events   details	
	2823-487-27720:13:11.8620000Z	
	2023-07-20709131:32.94500022   1	
	2023-06-17704:57:42.0530002	0.04
small	2823-82-861102:45:45.95800002 1   mag   lat   long   depth   sig   nst   title   cdi   msi   msgType   type   code	8.81
	2023-01-34113:43:13.2030002 1   mag   lat   long   depth sig   nst   title   cdi   mai   magfyps   type   code	
	2022-11-15711:53:19-58000022   1	
	time	
	mag   lat   long   depth   sig   nst   title   cdi   mmi   mag/lype   type   code	
	2023-88-05701:44:35.3060002	
5 pct	2023-07-28107-58-55.8960802	54.41
o por	2023-66-29723-43:39.81980802   1	01.11
	2023-65-28T04:21:28.9838082 1   mag lat   long   depth   sig   nst   title   cdi   mmi   magfype   type   code	
	2023-66-2115:24:43.5500002	
10 pct	time	
	4.7   -5.4883   34.8859   38.8   37.8   M 4.7 - Tercontin     mb   earthquake   78880rch	
	5.0   30.3389   94.7667   18.0   385   52.0   M 5.0 - 203 km N of Shi Vanit, Endia     nb   narthquake   70000pgp	
	mmg   lat   long   depth   sig   not   title   cit   mml   mmg/yee   type   code	130.05
	mag   lat   long   depth   sig   nst   title   cdi   mmi   mag/lyne   type   code	
	mag   lat   long   depth   sig   not   title   cdi   mmi   mag()ppe   type   code	
	mag   Jat   Jong   depth   sig   mst   title   cdi   mmi   mag/lype   type   code	

	+	
20 pct	Time   events   Code	295.83
30 pct	time   events	512.05
50 pct	Time	2416.42

	time   events   details	
	2023-08-27713:53:37.5640002	
80 pct	1.7   3.0000   3.0037   35.01   36.01   36.01   37.0	3438.75
	mag   lat   long   depth   sig   mst   title   cdi   msi   mag/lyne   type   code     4.7   42.3325   -29.2165   10.0   340   38.0   M 4.7 - Acores Islands region     mb   earthquake   70086qm5       2023-88-24705:35:24.685000Z       mag   lat   long   depth   sig   mst   title     cdi   mmi   mag/lyne   type   code	
	2023-08-24105:35:24.0850002 1   mag   lat   long   depth   sig   nst   title   cdi   mmt   magType   type   code       4.8   36.2229   38.1738   36.0   356   78.0   M A.S - 11 km SM of Yesllyurt, Turkey   4.6   mar   earthquake   7000Mogky	
	time events details  2023-08-27713:53:17.5640802 1   mag   lat   long   depth   sig   not   stitle   cell   mad   mag/lype   type   code     4.7   5.64080   34.8839   30.0   340   27.0   M 4.7 - Taxanda   sb   cartegade   70000rch	
	2023-08-20711:16:25.1840002	
large	5.1   58.582   348.8112   10.0   400   43.0   M.5.1 - west of Macquarie Island     now   earthquise   70006r5r	6724.65
	2023-08-19723:38-59-43000002	
	2023-08-19734-16:08-08000002	

Las gráficas con la representación de las pruebas realizadas.



### **Análisis**

A pesar de que obtener un elemento en un *arbo*l tiene una complejidad lineal O(N), la implementación incurre en ciclos anidados, lo cual lo puede hacer complejo,  $O(Y^2)$ .

Este comportamiento se puede evidenciar experimentalmente en la gráfica. Ya que, gracias a que los datos no se encuentran tan dispersos con respecto a la línea de tendencia, la curva coincide con el comportamiento exponencial esperado.

# **Requerimiento 4**

# Descripción

```
req_4(sig,gap,analyzer):
Función que soluciona el requerimiento 6
# TODO: Realizar el requerimiento
final = lt.newList('ARRAY_LIST')
newLista = lt.newList('ARRAY_LIST')
dic = \{\}
data_structs = analyzer['sig']
x = om.values(data_structs,float(sig), float(om.maxKey(data_structs)))
for i in lt.iterator(x):
    f = om.values(i,float(om.minKey(i)),float(gap))
    for j in lt.iterator(f):
        for z in lt.iterator(j):
            if len(z['gap'])>0:
             if float(z['gap'])>0:
                 1t.addFirst(newLista,z)
sa.sort(newLista,compareDates3)
a = lt.subList(newLista,1,17)
for z in lt.iterator(a):
    time = z['time']
    dic[time] = {
        'time':time,
        'events':1,
        'details':z
    lt.addLast(final,dic[time])
return final
```

Este requerimiento se encarga de o consultar los 15 eventos sísmicos más recientes ocurridos que superen una significancia mínima y que sean menores a una distancia azimutal indicada.

Entrada	• La significancia mínima del evento (sig). • La distancia azimutal máxima del evento (gap).
Salidas	El número total de eventos sísmicos registrados mayores a la significancia y menores a la distancia azimutal indicada • Los quince (15) eventos cronológicamente más recientes que cumplan con los parámetros especificados. Cada uno de los eventos en la consulta
Implementado (Sí/No)	Si. Implementado por Luis Sebastián Contreras

# Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	Complejidad
Creación de listas y diccionarios (tiempo constante):	O(1)
Esto incluye inicializar las estructuras de datos como	
listas y diccionarios.	
Buscar valores en un árbol binario	O(log(N))
Ciclo anidado for para recorrer la lista de valores	O(M)
Buscar valores en un árbol binario	O(log(M))
Ciclo anidado for para recorrer los valores	O(Y)
Verificación de condiciones y operaciones	O(1)
condicionales (tiempo constante)	
Ciclo anidado for para recorrer los valores	O(Y*Y)
Añadir al final de la lista los elementos	O(1)
Ordenamiento de la lista (sa.sort)	O(Y(log(Y))
TOTAL	O(Y*(Y))

# **Pruebas Realizadas**

Las pruebas realizadas fueron realizadas en una maquina con las siguientes especificaciones. Los datos de entrada fueron significancia mínima 300.0 y distancia azímutal máxima de 48.0.

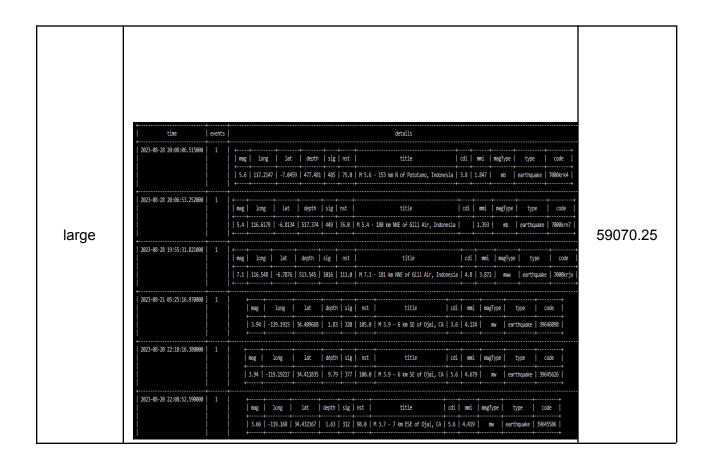
Procesadores	12th Gen Intel(R) Core(TM) i5-1235U	
Memoria RAM	16 GB	
Sistema Operativo	Windows 11 Home	

Entrada	Tiempo (ms)
small	10.58
5 pct	27.35
10 pct	77.78
20 pct	169.37
30 pct	283.64
50 pct	791.28
80 pct	12395.21
large	5907.25

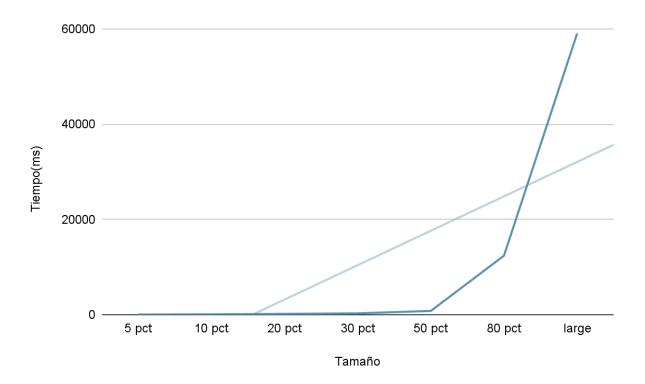
#### **Tablas de datos**

		Tiempo
Muestra	Salida	
small	time events	10.58
5 pct	time   events	27.35
10 pct	time   events	77.78

20 pct	time   conts	169.37
30 pct	time   events   Octails	283.64
50 pct	time   coeffs	791.28
80 pct	Table   Continue   C	12395.21



Las gráficas con la representación de las pruebas realizadas.



### **Análisis**

A pesar de que obtener un elemento en un *arbo*l tiene una complejidad lineal O(N), la implementación incurre en ciclos anidados, lo cual lo puede hacer complejo, O(Y^2).

Este comportamiento se puede evidenciar experimentalmente en la gráfica. Ya que, gracias a que los datos no se encuentran tan dispersos con respecto a la línea de tendencia, la curva coincide con el comportamiento exponencial esperado.

# **Requerimiento 5**

# Descripción

Este requerimiento se encarga de retornar los eventos sísmicos más recientes que superen una profundidad mínima y un número mínimo de estaciones de monitoreo. Sabemos que los datos están compuestos por un árbol el cual tiene una llave de profundidad y dentro del valor existe otro árbol que están el número mínimo de estaciones de monitoreo. Lo primero que hace es tomar los rangos de los dos árboles y ordenarlos de menor a mayor.

Entrada	La profundidad mínima del evento (depth). El número mínimo de estaciones que detectan el evento (nst).
Salidas	Un array list que cumpla con los requisitos anterios del mas reciente al mas antiguo.
Implementado (Sí/No)	Si. Implementado por Carlos Alberto Poveda Riaño

# Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	Complejidad
<pre>newLista = lt.newList('SINGLE_LINKED')</pre>	O(1)
Crea una lista encadenada	
<pre>x = om.values(data_structs,depth,</pre>	O(n)
om.maxKey(data_structs))	
Retorna todos los valores en una lista encadenada del	
arbol que se encuentren entre [keylo, keyhi]	
<pre>for i in lt.iterator(x):</pre>	O(n)
el valor de un arbol que es un arbol	
<pre>f = om.values(i,nst,om.maxKey(i))</pre>	O(n)
Retorna todos los valores en una lista encadenada del	
arbol que se encuentren entre [keylo, keyhi]	
<pre>for z in lt.iterator(j):</pre>	O(n)

<pre>lt.addLast(newLista,z )</pre>	
se agregan los elementos del arbol a una nueva lista	
<pre>merg.sort(newLista,compareDates2)</pre>	O(n log(n))
ordena los datos del mas reciente al mas antiguo	
TOTAL	O(n log(n))

# **Pruebas Realizadas**

Las pruebas realizadas fueron realizadas en una maquina con las siguientes especificaciones. Los datos de entrada fueron profundidad: 23 y numero de estaciones 38.

Procesadores	Intel(R) Core(TM) i5-6400 CPU
Memoria RAM	16 GB
Sistema Operativo	Windows 10

### Carga de datos

Entrada	Tiempo (ms)
small	1010.4
5 pct	5092.68
10 pct	11045.39
20 pct	20937.39
30 pct	34501.28
50 pct	59342.50
80 pct	95912.55
large	109279.5

### **Tablas de datos**

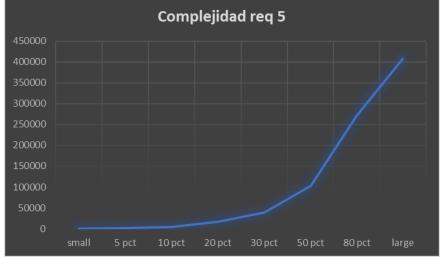
Muest	Tiempo	
Widest	Salida	

small	Table   Control   Fig.   Control	99.07
5 pct	Section of continue   1998	1142.09
10 pct	Section   Control   Cont	4250.64
20 pct	Section   Sect	16520.81

30 pct	Tests of eventor : 100002   Like   Secrets	39152.61
50 pct	Section   Control   Cont	102361.70
80 pct	Time	271223.89
large	Table   Second   Company   Company	407973.53

Las gráficas con la representación de las pruebas realizadas.





#### **Análisis**

Este requerimiento tiene una complejidad **O(n log(n))** a pesar de que utilizamos estructura de árboles para optimizar el requerimiento, necesita de ordenamientos que influencia en la complejidad, al inicio tiene una complejidad constante, sin embargo, al acceso de los valores de los árboles y los ordenamientos tiende la complejidad a aumentar, De manera que se evidencia que la complejidad en el mejor va ser siempre **O(n log(n))**.

Este comportamiento se puede evidenciar experimentalmente en la gráfica. Ya que, gracias a que los datos no se encuentran tan dispersos con respecto a la línea de tendencia, la curva coincide con el comportamiento curvo esperado.

# Requerimiento 6

# Descripción

```
def req_5(year,lat,lon,radio, data_structs):
   max = \{\}
   c = lt.newList('SINGLE_LINKED')
   array = lt.newList('ARRAY_LIST')
   array2 = lt.newList('ARRAY_LIST')
   temblor =m.get(data_structs,year)
   temblores = me.getValue(temblor)
    for j in lt.iterator(temblores):
        distancia = getdistance(lon,lat,j['long'],j['lat'])
       time = j['time']
       time2 = datetime.datetime.strptime(j['time'],'%Y-%m-%dT%H:%M:%S.%fZ')
        j['time'] = time2
        j['distancia'] = round(distancia,3)
        if j['distancia'] <radio:</pre>
            lt.addLast(array,j)
            if a <float(j['mag']):
    a =float(j['mag'])</pre>
                max = j
   lt.addLast(c,max)
    f = merg.sort(array,compareDates3)
   return f, c
```

Este requerimiento se encarga de retornar el sismo más significativo de un año dado dentro de un área circundante de una coordenada GPS designada, y los N eventos sísmicos más próximos cronológicamente. Se toma el valor del año de una tabla hash y se hacen los respectivos cálculos para obtener la distancia y así organizarlos.

Entrada	El año relevante (en formato "%Y").  La Latitud de referencia para el área de eventos (lat).  La longitud de referencia para el área de eventos (long).  El radio [km] del área circundante (float).  El número de los N eventos de magnitud más cercana a mostrar.
Salidas	el evento maximo y un array list de los temblores mas recientes
Implementado (Sí/No)	Si. Implementado por Carlos Alberto Poveda Riaño

# Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	Complejidad
<pre>array = lt.newList('ARRAY_LIST')</pre>	O(1)
Se crea un array list	
<pre>temblor =m.get(data_structs,year)</pre>	O(1)
Retorna la pareja llave, valor, cuya	
llave sea igual a key	

<pre>temblor =m.get(data_structs,year)</pre>	O(1)
Retorna el valor de una pareja de un	
Map	
<pre>for j in lt.iterator(temblores):</pre>	O(n)
<pre>merg.sort(array,compareDates3)</pre>	O(n(log(n))
<pre>lt.addLast(array,j)</pre>	O(1)
agrega el elemento a la lista	
TOTAL	O(n(log(n)))

# **Pruebas Realizadas**

Las pruebas realizadas fueron realizadas en una maquina con las siguientes especificaciones. Los datos de entrada fueron año: 2022, latitud: 4.674, longitud: 74.068, radio: 3000, numero de eventos: 5

Procesadores	Intel(R) Core(TM) i5-6400 CPU
Memoria RAM	16 GB
Sistema Operativo	Windows 10

Entrada	Tiempo (ms)
small	1010.4
5 pct	5092.68
10 pct	11045.39
20 pct	20937.39
30 pct	34501.28
50 pct	59342.50
80 pct	95912.55
large	109279.5

### **Tablas de datos**

		Tiemp
Muestra	Salida	

small	The control of the	4.64
5 pct	Table	19.03
10 pct	The control of the	39.15
20 pct	The control of the	78.34
30 pct	Column   C	116.64

50 pct	March 2018   1986   1	195.89
80 pct	March   Marc	316.86
large	March   Marc	394.41

Las gráficas con la representación de las pruebas realizadas.



### **Análisis**

En este requerimiento se utilizó la tabla hash lo que permitió la accesibilidad de los datos más rápido, aunque el ordenamiento tiene un orden lineal O(n). Esto debido a que, lo primero que se hace una búsqueda en la tabla de hash si el elemento hace parte del mapa. Sin embargo, al ordenarlo y al hacer un ciclo la complejidad es lineal.

Este comportamiento se puede evidenciar experimentalmente en la gráfica. Ya que, gracias a que los datos no se encuentran tan dispersos con respecto a la línea de tendencia, la curva coincide con el comportamiento lineal esperado.

# **Requerimiento 7**

# Descripción

```
def req_7_histogram(year, title, prop, bins, analyzer):
   prop_values = lt.newList('ARRAY_LIST')
   year = me.getValue(m.get(analyzer['year'],int(year)))
   lista = lt.newList('ARRAY_LIST')
   prop_values2 = []
   for date in lt.iterator(year):
       if title in date['title']:
           if date[prop] is not None:
             lt.addLast(prop_values, date[prop])
             lt.addLast(lista, date)
   sa.sort(prop_values, compare_prop)
   sa.sort(lista, compareDates2)
   for a in lt.iterator(prop_values):
       prop_values2.append(a)
   mayor = lt.firstElement(prop_values)
   menor = lt.lastElement(prop_values)
   return lt.size(year), lt.size(lista), mayor, menor, prop_values2, lista
```

Este requerimiento se encarga de contabilizar los eventos sísmicos ocurridos en una región y un año especifico según alguna propiedad de interés como lo son su magnitud (mag), profundidad (depth) o la significancia del evento (sig).

Entrada	•El año relevante (en formato "%Y"). • El título de la región
	asociada ("title"). • La propiedad de conteo (magnitud, profundidad
	o significancia). • El número de segmentos o casillas (bins) en los
	que se divide el histograma.

Salidas	El número de eventos sísmicos dentro del periodo anual relevante.
	• El número de eventos sísmicos utilizados para crear el histograma
	de la propiedad. • Valor mínimo y valor máximo de la propiedad
	consultada en el histograma. • El histograma con la distribución de
	los eventos sísmicos según la propiedad. • Listado de los eventos
	que cumplen las condiciones de conteo para el histograma.
Implementado (Sí/No)	Si. Implementado por Luis Sebastián Contreras

# Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	Complejidad		
Inicialización de estructuras de datos y conversiones	O(1)		
de fechas:			
Obtención de elementos según rango de fechas	O(N)		
Ciclo anidado for para recorrer la lista de valores	O(N)		
Creación y adición de elementos a una lista:	O(1)		
Ordenar los elementos de la lista	O(Mlog(M))		
Iterar los elementos de la lista	O(M)		
Añadir a la lista un elemento	O(1)		
TOTAL	O(M(log(M))		

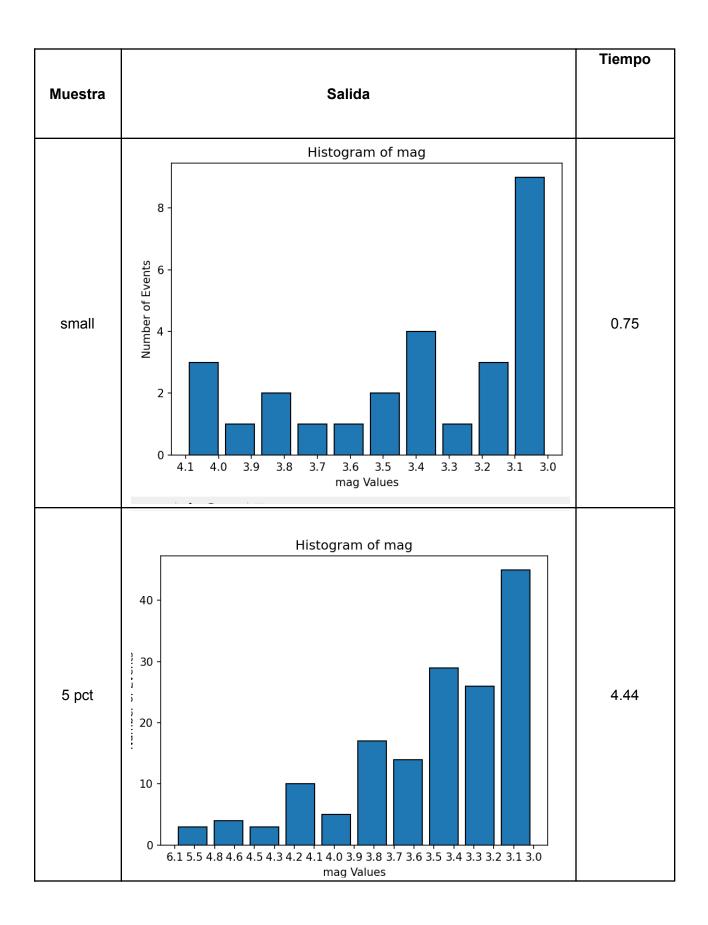
# **Pruebas Realizadas**

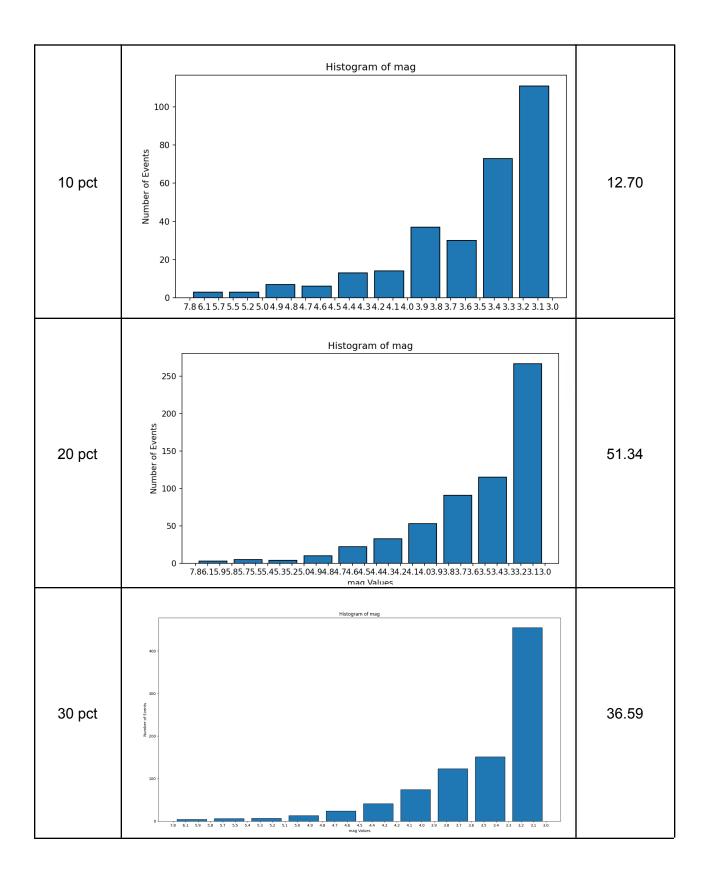
Las pruebas realizadas fueron realizadas en una maquina con las siguientes especificaciones. Los datos de entrada fueron significancia mínima 300.0 y distancia azímutal máxima de 48.0.

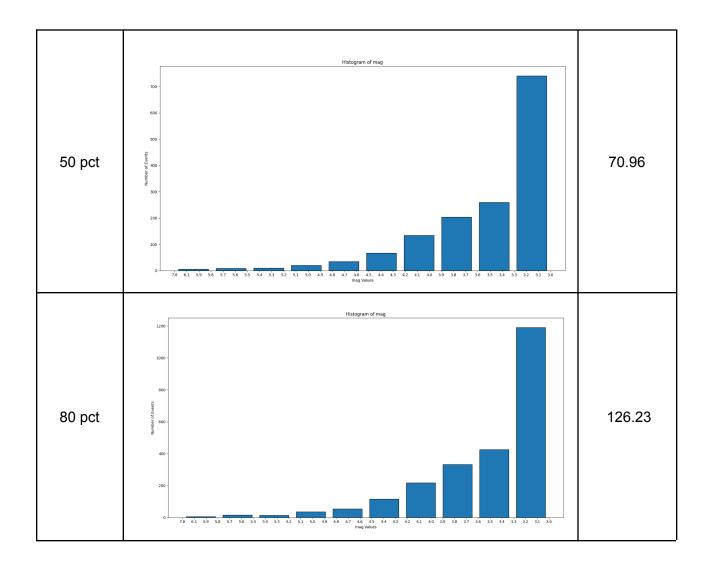
Procesadores	12th Gen Intel(R) Core(TM) i5-1235U		
Memoria RAM	16 GB		
Sistema Operativo	Windows 11 Home		

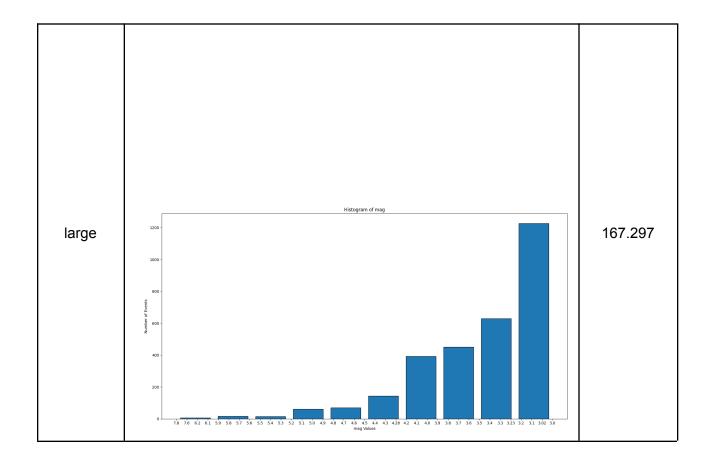
Entrada	Tiempo (ms)
small	0.75
5 pct	4.44
5 pct 10 pct	12.70
20 pct	51.34
30 pct	36.59
50 pct	70.96
80 pct	126.23
large	167.297

# **Tablas de datos**









Las gráficas con la representación de las pruebas realizadas.

# **Análisis**

La obtención de los elementos en un Hashmap puede resultar eficiente al tener una complejidad de N, sin embargo para poder obtener los datos correctamente se debe hacer un ordenamiento, esto puede conllevar a un complejidad de MlogM. Sin embargo, esto no refleja un cambio muy contundente, en

términos		generales	el	С	ódigo	es	eficiente.
200 —							
150 —							
100 —							
50 —							
0 —	5 pct	10 pct	20 pct	30 pct	50 pct	80 pct	 large