ANÁLISIS DEL RETO

Nicolás Torres Pulido (Est 1) Cod 202123826
Andres Camilo Bonilla (Est 2) Cod 202226897
Juan Esteban Vásquez (Est 3) Cod 202220053

Carga de datos

Procesadores	**Intel(R) Core(TM) i3-10110U CPU @ 2.10GHz 2.59 GHz**

Memoria RAM (GB)	8.0
Sistema Operativo	Windows 11 Home – 64 bits

Carga de Catálogo PROBING

Factor de Carga (PROBING)	Consumo de Datos [kB]	Tiempo de Ejecución Real @LP [ms]
0.1	21777.25	13360.73
0.5	21777.25	13468.47
0.7	21777.25	13510.43
0.9	21777.25	13561.91

Carga de Catálogo CHAINING

Factor de Carga (CHAINING)	Consumo de Datos [kB]	Tiempo de Ejecución Real @SC [ms]
2.00	21779.06	13259.96
4.00	21779.06	13362.21
6.00	21779.06	13392.48
8.00	21779.06	13411.22

Requerimiento 1

Descripción

El requerimiento número 1, se encarga de obtener la actividad económica con mayor saldo a pagar para un sector económico y un año especifico. Para lograr esto, primero le pide a través de un input al usuario que escriba el código del sector económico y el año que desea obtener, estos se envían por

parámetro para posteriormente ser enviado al controlador y después al modelo. Donde inicializamos 4 variables una con un str vacío, otra con un cero, otra con la información del ano extraída del mapa y la última contiene toda la lista que viene en ese año a través de la función lt.iterator(). Después hace uso de un bucle for que recorre estos datos hasta encontrar un elemento que coincida con el código que el usuario paso por parámetro para posteriormente ir comparando cual es el que tiene un mayor saldo a pagar mediante un if y una vez termina el proceso lo envía a la vista para imprimirlo.

Entrada	control, year, id (código del sector económico)
Salidas	La actividad económica con mayor saldo a pagar para un sector económico y un año especifico.
Implementado (Sí/No)	Sí

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Inicializar variables: actividad, saldo_ap, info_ano	O(1)
Inicializar varibles: datos	O(N)
For de datos	O(N)
TOTAL	***O(N)***

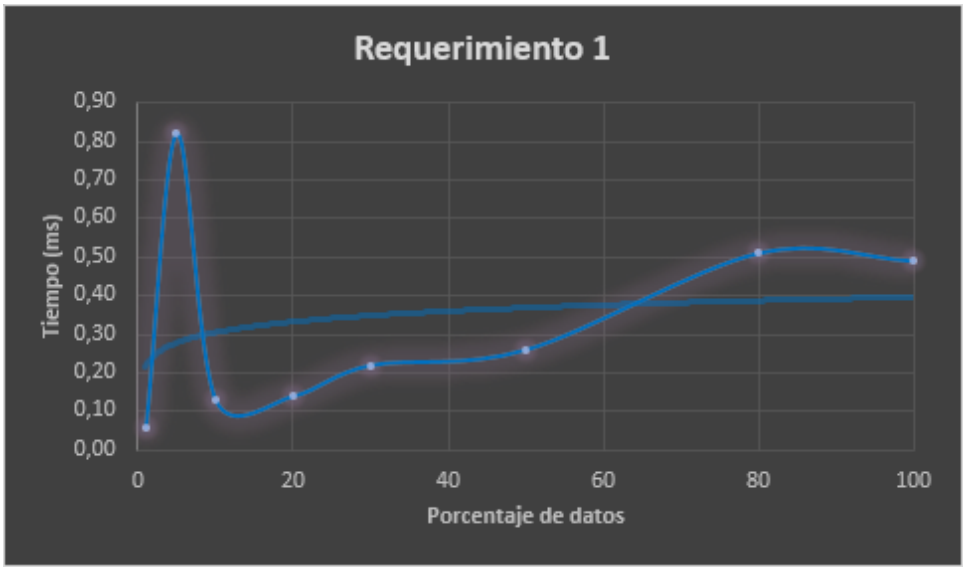
Pruebas Realizadas

Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

Procesadores	**Intel(R) Core(TM) i7-8565U**
Memoria RAM	8 GB
Sistema Operativo	Windows 10

Entrada	**Tiempo (s)**
small	0.06
5 pct	0.82
10 pct	0.13
20 pct	0.14
30 pct	0.22
50 pct	0.26
80 pct	0.51
large	0.49

Graficas



Análisis

En cuanto al requerimiento uno si se compara con el anterior los resultados son más rápidos, ya que la estructura de datos que se usa es mucho más rápida que las listas del reto 1, podemos ver que sin tener en cuenta el uso de memoria los tiempos de carga en diferentes tipos de tamaños es bastante poco y no varía mucho. En cuanto a la complejidad, ambos tienen O(n), dado que recorren todos los datos en el peor de los casos.

Requerimiento 2

Descripción

El requerimiento numero dos se encarga de obtener la actividad económico con mayor saldo a favor para un sector económico y un año específico. Para lograr esto en la vista se le pide al usuario a través de dos inputs tanto el año como el código del sector económico, enviándolos al modelo, en donde se inicializan 5 variables y después con un bucle for se recorren los datos de del año que el usuario coloco anteriormente, si existe el un elemento que su código coincida con el que coloco el usuario, entonces mediante un if se comparan los valores hasta obtener los outputs, que terminan siendo retornados por la funcion.

Entrada	control, year, id (código del sector económico).
Salidas	la actividad económica con mayor saldo a favor para un sector económico y un año específico.
Implementado (Sí/No)	Si

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Inicializar variables: actividad, saldo_af, info_ano	O(1)
Inicializar varibles: datos	O(N)
For de datos	O(N)
TOTAL	***O(N)***

Pruebas Realizadas

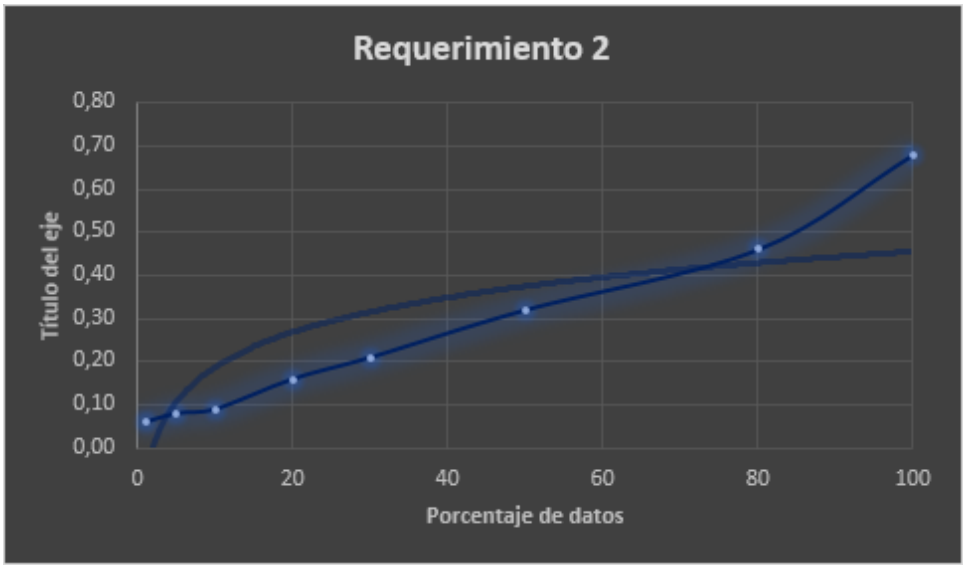
Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

Procesadores	**Intel(R) Core(TM) i7-8565U**
Memoria RAM	8 GB
Sistema Operativo	Windows 10

Entrada	**Tiempo (s)**
small	0.06
5 pct	0.08
10 pct	0.09
20 pct	0.16
30 pct	0.21

50 pct	0.32
80 pct	0.46
large	0.68

Graficas



Análisis

Los tiempos de ejecución del requerimiento dos mejoran considerablemente gracias a la estructura de los mapas que simplemente hace que la complejidad temporal en el peor de los casos sea de O(N).

Requerimiento 3

Descripción

El requerimiento número 3, se encarga de que a partir de un año especifico encuentra el subsector económico con el menor total de retenciones. Para esto primero en el view.py se le pide al usuario el año el cual desea buscar, el cual lo pasamos por parámetro y lo enviamos hasta el model.py, donde creamos un mapa con la función newMap(), después recorremos todos los datos de la data a partir de ese año y si el elemento ya existe en el mapa simplemente actualizamos los valores y en caso de que no exista lo creamos. Una vez terminado este proceso la función se encarga de encontrar el elemento que tenga el menor total de retenciones y por lo tanto lo retorna.

Entrada	control, year
Salidas	El subsector económico con el menor total de retenciones
Implementado (Sí/No)	Sí. Andrés Camilo Bonilla.

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Crear un nuevo mapa (newMap)	O(1)
Recorrer la data mediante un for	O(n)
Funcion que ordena la data con un sort (sort)	O(nlogn)
TOTAL	***O(nlogn)***

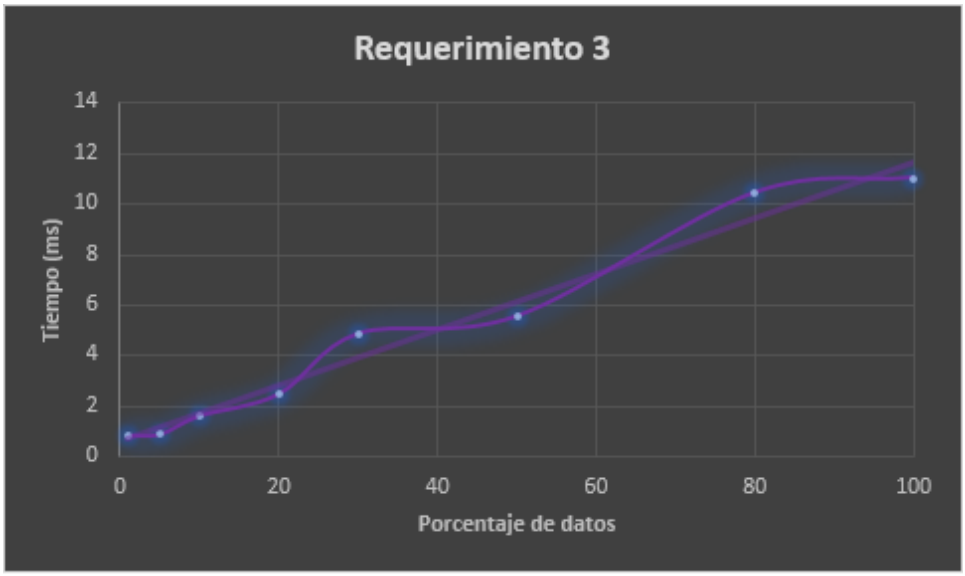
Pruebas Realizadas

Las pruebas realizadas fueron realizadas en una maquina con las siguientes especificaciones.

Procesadores	**Intel(R) Core(TM) i7-8565U**
Memoria RAM	8 GB
Sistema Operativo	Windows 10

Entrada	**Tiempo (ms)**
small	0.85
5 pct	0.88
10 pct	1.61
20 pct	2.48
30 pct	4.87
50 pct	5.56
80 pct	10.44
large	11.03

Graficas



Análisis

La mejora en el tiempo de este función y requerimiento es considerablemente mejor respecto a la anterior, ya que la complejidad es de O(N), ya que primero es mucho más fácil acceder a los datos del mapa y al querer encontrar un elemento simplemente tiene una complejidad O(1)

Requerimiento 4

Descripción

Para este problema se aprovechó que la carga de datos se hizo en un mapa ordenado por años, lo que permitió usar el año recibido como parámetro como llave para obtener las actividades de dicho año. De esta forma, hubo que recorrer estas actividades para hallar todos los subsectores y guardarlos como llaves en un mapa, donde cada uno tenía como valor un arreglo con su información, totales y todas sus actividades. Al ir iterando también se iba guardando en una variable auxiliar el código del subsector que tuviera mayores costos y gastos de nómina, para que una vez finalizara el ciclo, se pudiera llamar dicho subsector y ordenar sus actividades por costos y gastos de nómina y así construir las tablas asociadas a la información de las actividades del subsector

Entrada	Mapa de la información de la DIAN ordenada por años y el año a consultar
Salidas	Un arreglo con la información del subsector con mayores costos y gastos en nómina y sus correspondientes actividades económicas
Implementado (Sí/No)	Sí. Nicolás Torres.

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Creación de nuevo mapa	O(1)
Selección de datos del año seleccionado por parámetro	O(1)
Recorrer los datos del año con un (en el peor de los casos todos los datos serán del mismo año)	O(n)
Creación de lista como valor de cada subsector como llave con la función put y asignación de su correspondiente información inicial inicial	O(n)
Obtener el arreglo con la función get para sumar los items solicitados	O(n)
Comparación para hallar el subsector con mayor total de costos y gastos de nómina	O(n)
Guardar la actividad en el arreglo	O(n)
Obtener el subsector con mayores costos y gastos de nómina dentro del mapa	O(1)
Sortear las actividades económicas del subsector de acuerdo a su total de costos y gastos de nómina	O(nlogn)
TOTAL	***O(n)***

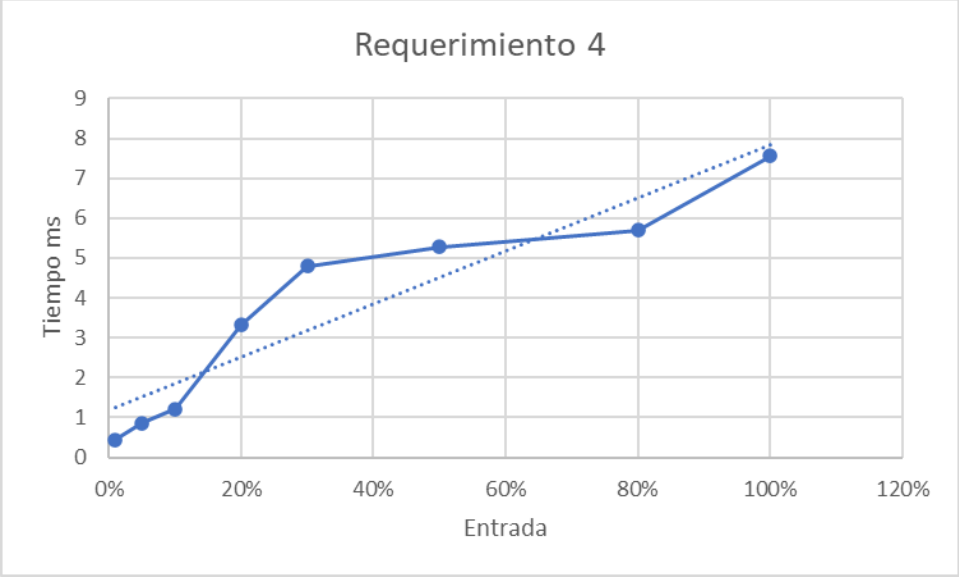
Pruebas Realizadas

Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

Procesadores	Intel(R) Core(TM) i3-10110U CPU @ 2.10GHz 2.59 GHz
Memoria RAM	8.0
Sistema Operativo	Windows 11 Home
Año	2017

Entrada	**Tiempo (ms)**
small	0,44
5 pct	0,86
10 pct	1,21
20 pct	3,32
30 pct	4,8
50 pct	5,28
80 pct	5,69
large	7,56

Graficas



Análisis

El desarrollo de este requerimiento permitió evidenciar una notable mejoría en el orden de complejidad de la función, pues gracias al uso de los mapas es mucho más fácil y sencillo acceder a información de acuerdo a un parámetro y de esta forma evitar tener que hacer múltiples recorridos. En este caso se consiguió una complejidad de O(n), pues con la información ordenada por años, solo es necesario acceder al valor que tiene ese año como llave y en el peor de los casos, todos los datos pertenecerían a

este mismo año. Otro aspecto importante es la mejoría en el tiempo, dado que los mapas permiten acceder a información específica en un orden de O(1), por lo que, una vez identificada la información necesaria, solo hay que conseguir su llave, en lugar de recorrer todo nuevamente. A comparación del reto 1, los tiempos de ejecución son mucho más veloces a pesar de que en ambos casos la complejidad sea O(n). Esto se puede explicar gracias a como el uso de los mapas fácilita el acceso a información específica,

Requerimiento 5

Descripción

El requerimiento 5 esencialmente pide un año de búsqueda y debe devolver el subsector económico para el año especificado para el que la suma de descuentos tributarios sea mayor. Esto se hace por medio de dos búcles en donde se organiza la información y después se distribuye acorde a la comparación deseada.

Entrada	(control) Catálogo de datos de la DIAN y (year) el año de búsqueda
Salidas	Tabla con la información del subsector de mayores descuentos tributarios y la información de las tres menores o mayores actividades en descuentos para el subsector (si hay al menos 6).
Implementado (Sí/No)	Sí. Juan Esteban Vásquez.

Análisis de complejidad

Pasos	**Complejidad**
Asignación de 'nm', 'yearInfo' y luego de 'most'	O(1)
Búcle de for en el que se recorren las entradas	O(N)
Asignación de la variable 'subs' con el nombre de los subsectores del año	O(N)
Condicion mp.contains() y el lt.isPresent() en bucket / recorrido de la lista (con un factor de carga eficiente)	O(N)
Función mp.put() y la creación de la lista nueva	O(N)
Encontrar valor actual y añadirle información del elemento actual a la lista al final	O(N)
Aumentar el contador de las variables del subsector	O(N)
Búcle que recorre las llaves del diccionario	O(N)
Asignación subsector actual y comparación de mayores descuentos tributarios, y reasignación en caso de True.	O(N)
Función que sortea todas las listas de los subsectores	O(NlogN)
TOTAL	***O(NlogN)***

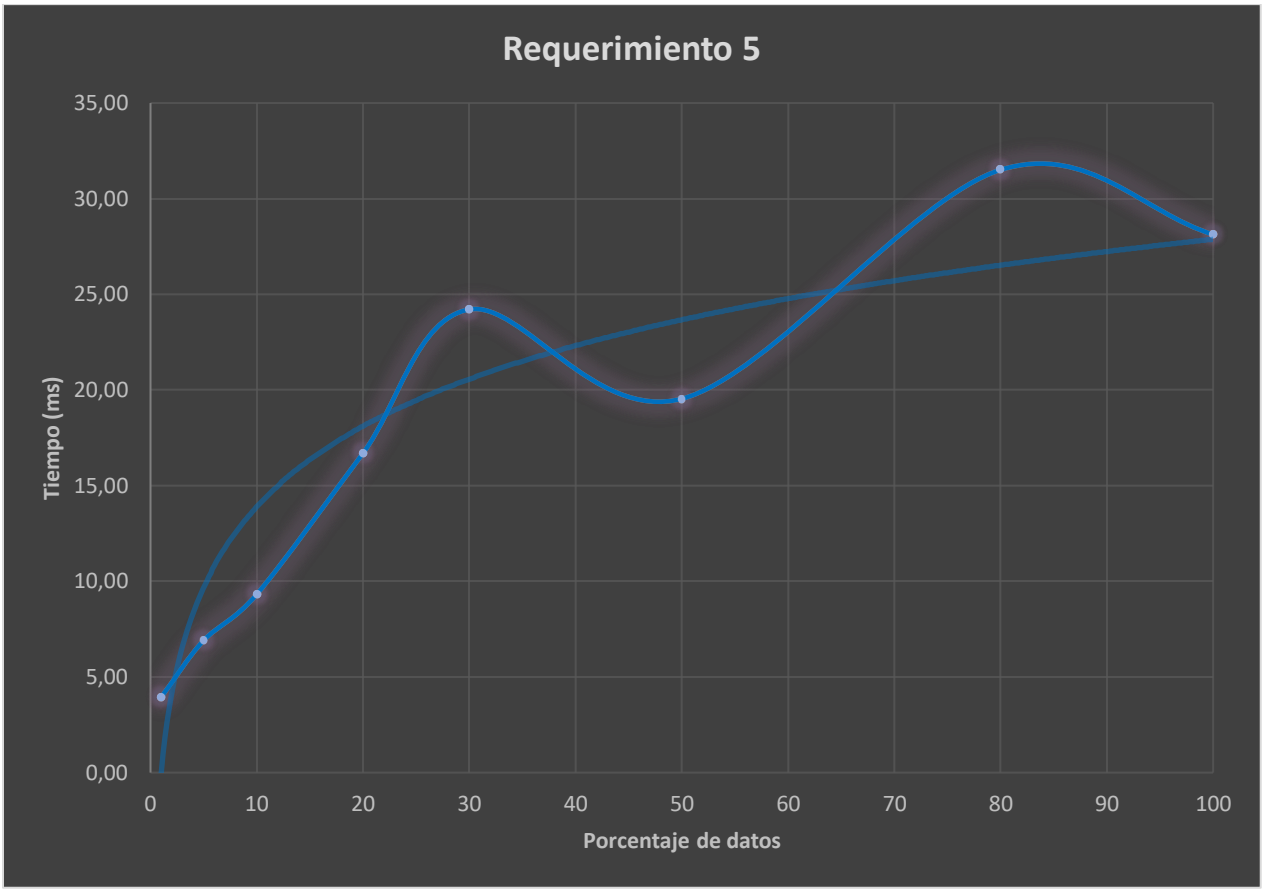
Pruebas Realizadas

Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

Procesadores	M2
Memoria RAM	8 GB
Sistema Operativo	MacOS Ventura 13.2
Año de Búsqueda	2020

Entrada	Tiempo (ms)
small	3.94
5 pct	6.91
10 pct	9.32
20 pct	16.70
30 pct	24.21
50 pct	19.54
80 pct	31.51
large	28.13

Graficas



Análisis

En comparación con el reto anterior, es posible decir que la complejidad en notación O es menor gracias a la manera de obtener los datos desde un mapa, comparado a una lista. Si este mismo requerimiento se hace a partir de listas, es muy probable que el tiempo sería mayor. De igual manera, es posible decir que el tiempo promedio que este proceso se demora en el peor de los casos (en que todos los datos iniciales sean del año especificado y que además todos ahgan parte del mismo subsector económico) este proceso se demoraría O(N) gracias a que se recorrerían todos los datos en el búcle incial y en el sorteo de la lsita del subsector de mayores descuentos tributarios. Con respecto al reto 1 hubo una

notable mejora tanto en tiempo como en orden de complejidad, pues se pasó de un O(n^3) a O(n) gracias a como el uso de los mapas permite evitar hacer ciclos dentro de ciclos.

Requerimiento 6

Descripción

El requerimiento 6 debe sumar los totales de ingresos netos para todas las actividades económicas del mismo subsector para devolver el subsector en que se de el mayor total para el año especificado por parámetro.

Entrada	(control) Catálogo de datos de la DIAN y (year) el año de búsqueda
Salidas	Tabla con la información del subsector de mayores ingresos netos y la información de las tres menores o mayores actividades en descuentos para el subsector (si hay al menos 6).
Implementado (Sí/No)	Sí.

Análisis de complejidad

Pasos	**Complejidad**
Asignación inicial de variables y creación de un nuevo mapa. (‘mapa’, ‘sector’, ‘secmax’)	O(1)
Obtención de los datos de un año especifico	O(1)
Crea una lista iterativa para los datos del año correspondiente.	O(N)
Búcle for para recorrer la lista de datos anterior	O(N)
Asignación de variables para nombre y código del subsector	O(N)
Comparación de contención de un subsector en el mapa (con factor de carga apropiado)	O(N)
Posicionamiento de variables contadoras en el valor del mapa principal	O(N)
Adición de valores a variables contadoras.	O(N)
Comparación con valor variable existente de mayor y menor y reemplazo en caso de True.	O(N)
Obtención de valores de los mapas	O(1)
Funciones de sorteo (que en el peor caso, ambas tienen los datos totales)	O(NlogN)
Obtener el valor del sector final	O(1)
TOTAL	***O(NlogN)***

Pruebas Realizadas

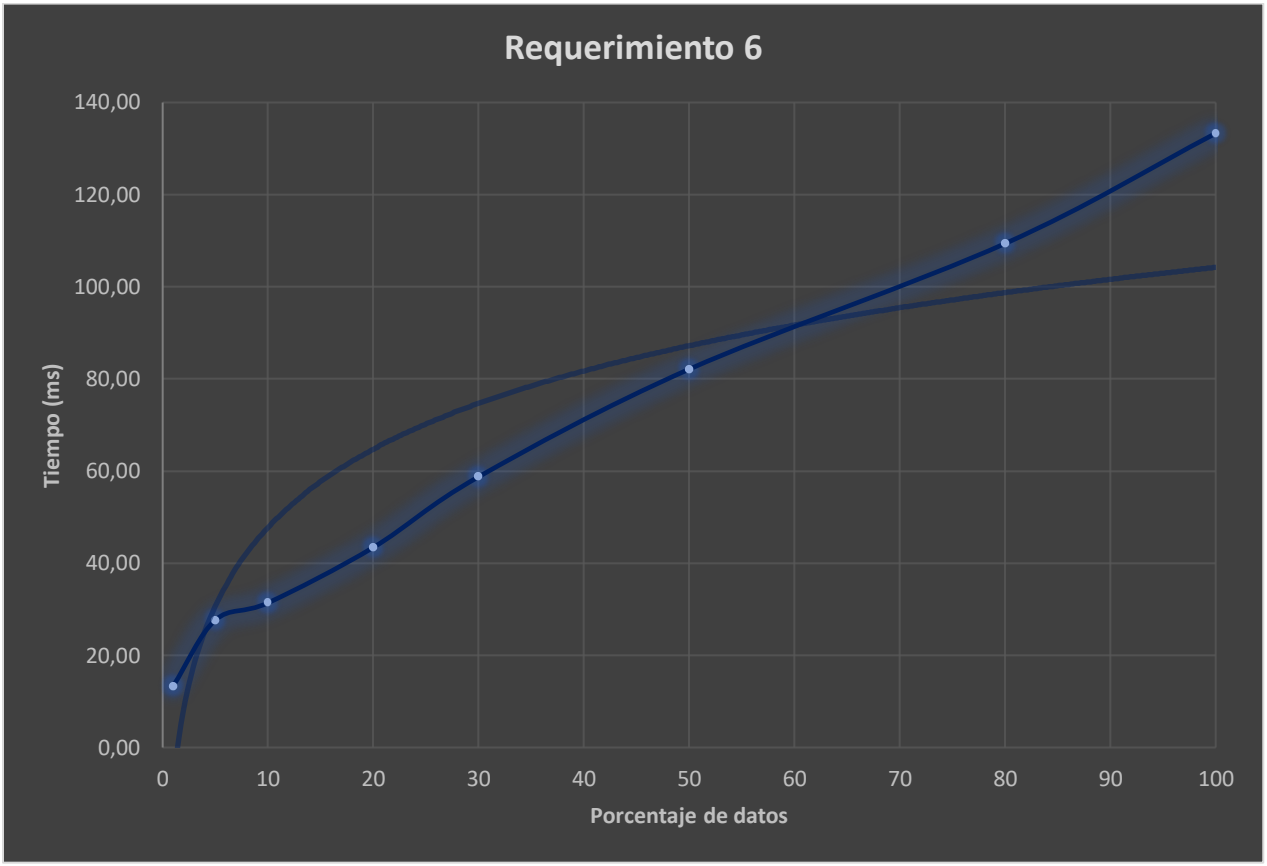
Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

Procesadores	M2
Memoria RAM	8 GB
Sistema Operativo	MacOS Ventura 13.2

Año de Búsqueda	2020

Entrada	**Tiempo (ms)**
small	13.35
5 pct	27.66
10 pct	31.40
20 pct	43.42
30 pct	58.75
50 pct	82.11
80 pct	109.38
large	133.30

Graficas



Análisis

Podemos observar que en este caso, la función hace uso de varios mapas para crear una organización de la información, tal que existan mapas dentro de mapas para organizar la información de esta manera. Si pensamos en la utilización de listas para lograr este mismo propósito, podemos pensar que se demoraría relativamente más tiempo en encontrar objetos, por ejemplo, dado que se debería recorrer todos los elementos en las maneras en que están organizados para obtener información que se desea. Esto, gracias a que no se usa el mismo método de llaves y hashValues para los mapas. Nuevamente hubo una notable mejora, no solo en tiempo de ejecución sino en orden de complejidad. Gracias al uso

de mapas fue posible pasar de un O(n^2) a O(nlogn) gracias a la forma en que se puede acceder fácilmente a información por medio de los mapas, lo que ayuda a evitar tener que hacer múltiples bucles.

Requerimiento 7

Descripción

El requisito 7 busca y muestra un top de actividades económicas que tienen un menor total acumulado de costos y gastos para un año y un subsector económico específico. Para hacer esto mismo, se organizan los datos de tal manera que si una actividad pertenece al año y subsector especificados, se añade a una lista de tipo arreglo, la cual después organiza sus datos basado en el criterio (mayor total de costos y gastos) y devuelve la lsita ordenada, la cual imprime el número de datos que se especifiquen. Para este requisito se utilizaron tanto el TAD de mapas, como el de listas, por lo que la implementación logró ser relativamente simple y corta.

Entrada	Parámetros necesarios para resolver el requerimiento.
Salidas	Respuesta esperada del algoritmo.
Implementado (Sí/No)	Sí.

Análisis de complejidad

Pasos	**Complejidad**
Crear una variable vacía que contenga un Arreglo	O(1)
Obtiene la información del año de búsqueda	O(1)
Crea una lista iterable con base en la información anterior	O(N)
Búcle que recorre la lista iterada anterior y asigna el valor del código del subsector actual a una variable.	O(N)
Comparación de código de entrada y variable asignada anteriormente.	O(N)
Adición del dato de subsector en caso de True	O(N)
Función de sorteo de la lista final de actividades	O(NlogN)
TOTAL	***O(NlogN)***

Pruebas Realizadas

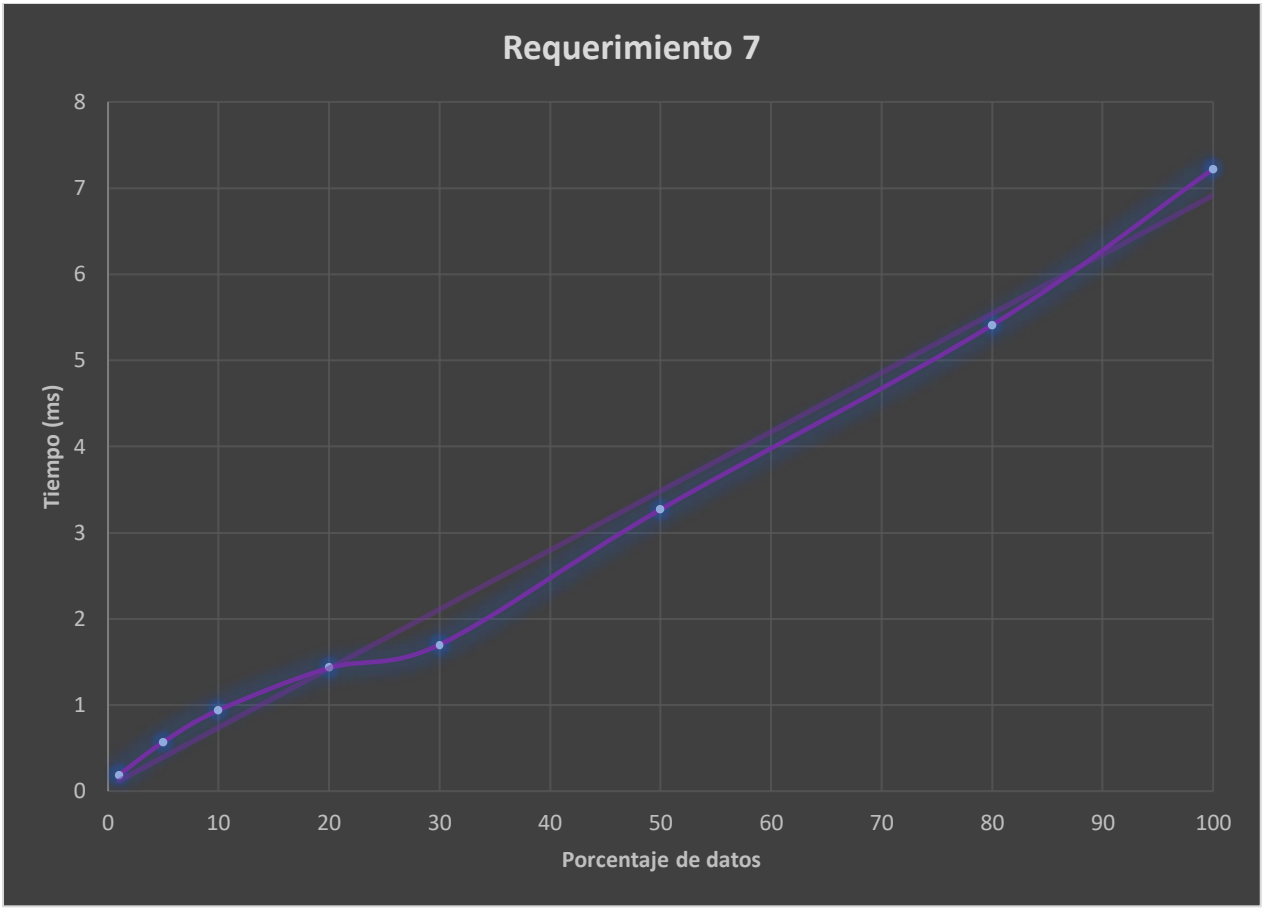
Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

Procesadores	M2
Memoria RAM	8 GB
Sistema Operativo	MacOS Ventura 13.2
Año de Búsqueda	2020
Código del subsector a buscar	11
Actividades a listar	5

Entrada	**Tiempo (ms)**

small	0.19
5 pct	0.57
10 pct	0.94
20 pct	1.43
30 pct	1.70
50 pct	3.27
80 pct	5.41
large	7.22

Graficas



Análisis

En esta función se hace uso de ambas estructuras de dato para aumentar la eficiencia del requerimiento. En este caso, si se tratara únicamente del TAD list, este proceso, así como los pasados, se demoraría más en los peores casos, dado que la obtención inicial de los datos del año de búsqueda se demoraría en el peor de los casos O(N), lo cuál afectaría únicamente el inicio de la función, pero que tendría un impacto en cantidades mucho mayores de datos. Gracias al uso de mapas fue posible pasar de un O(n^2) en el reto anterior a O(nlogn) gracias a la forma en que se puede acceder fácilmente a información por medio de los mapas, lo que ayuda a evitar tener que hacer múltiples bucles.

Requerimiento 8 (bonus)

Descripción

Para el desarrollo de este requerimiento se siguió la misma lógica que en los individuales, al guardar la información de cada subsector y sus totales en un mapa para así poder acceder a ella después fácilmente por medio de las llaves. Una vez recorridos todos los datos, se obtuvieron las llaves del mapa y se usaron para extraer cada subsector y colocarlo en un arreglo. Una vez realizado esto, se sortearon las actividades de cada subsector de acuerdo a los impuestos a cargo y posteriormente se hizo lo mismo a gran escala con los subsectores de acuerdo a total de impuestos a cargo. Esto permitió que se pudiera obtener el top N deseado por medio de llamar elementos del arreglo con un for en un rango desde 1 hasta N +1.

Entrada	El mapa con la información de la DIAN ordenada por años, el año de consulta ingresado por parámetro y el valor de N para la realización del top N de subsectores también ingresado como parámetro
Salidas	Un arreglo con los subsectores ordenados de acuerdo s sus impuestos a cargo
Implementado (Sí/No)	Sí.

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Crear mapa	O(1)
Recorrer el mapa recibido como parámetro en la llave del año recibido como parámetro	O(n)
Creación del subsector como llave en el nuevo mapa con valor de un arreglo	O(n)
Acceder a la llave del subsector para guardar y sumar su información	O(n)
Guardar la actividad económica dentro del arreglo de su subsector económico	O(n)
Obtener todas las llaves del mapa	O(1)
Recorrer las llaves del mapa	O(n)
Tomar el arreglo de cada subsector y sortearlo con merge de acuerdo a los impuestos a cargo de las actividades	O(nlogn)
Sortear los subsectores de acuerdo a sus impuestos a cargo	O(nlogn)
TOTAL	***O(n^2)***

Pruebas Realizadas

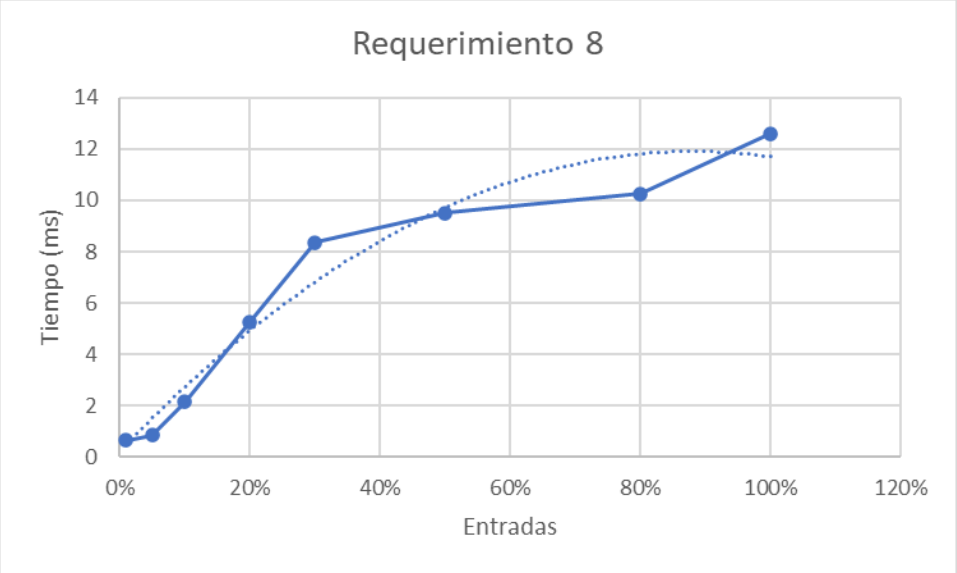
Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

Procesadores	Intel(R) Core(TM) i3-10110U CPU @ 2.10GHz 2.59 GHz
Memoria RAM	8 GB
Sistema Operativo	Windows 11 Home
Año de Búsqueda	2017
N	25

Entrada	**Tiempo (ms)**
small	0,65
5 pct	0,84
10 pct	2,13
20 pct	5,27
30 pct	8,36
50 pct	9,5
80 pct	10,27
large	12,6

Graficas

Las gráficas con la representación de las pruebas realizadas.



Análisis

El comportamiento observado corresponde a un orden exponencial dado que no solo hay que recorrer los datos para acomodarlos en cada subsector, sino que al finalizar el primer ciclo es necesario llamar la totalidad de los subsectores, (que en el peor de los casos podrían ser N), ordenar sus actividades

económicas y luego ordenar los propios subsectores dentro de su arreglo para así poder sacarlos en orden para construir la tabla del Top N. SIn embargo, el uso de mapas permite realizar la extracción de la información de los mapas de forma rápida, lo que reduce los tiempos de ejecución notablemente y permite que incluso con el archivo más grande, el requerimiento funcione en cuestión de pocos milisegundos. Gracias al uso de mapas fue posible pasar de un O(n^2) en el reto 1 a O(nlogn) gracias a la forma en que se puede acceder fácilmente a información por medio de los mapas, lo que ayuda a evitar tener que hacer múltiples bucles.

Requerimiento Ejemplo

Descripción

```python
def get_data(data_structs, id):
    """
    Retorna un dato a partir de su ID
    """
    pos_data = lt.isPresent(data_structs["data"], id)
    if pos_data > 0:
        data = lt.getElement(data_structs["data"], pos_data)
        return data
    return None
```

Este requerimiento se encarga de retornar un dato de una lista dado su ID. Lo primero que hace es verificar si el elemento existe. Dado el caso que exista, retorna su posición, lo busca en la lista y lo retorna. De lo contrario, retorna None.

Entrada	Estructuras de datos del modelo, ID.
Salidas	El elemento con el ID dado, si no existe se retorna None
Implementado (Sí/No)	Si. Implementado por Juan Andrés Ariza

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Buscar si el elemento existe (isPresent)	O(n)
Obtener el elemento (getElement)	O(1)
TOTAL	***O(n)***

Pruebas Realizadas

Las pruebas realizadas fueron realizadas en una maquina con las siguientes especificaciones. Los datos de entrada fueron el ID 1.

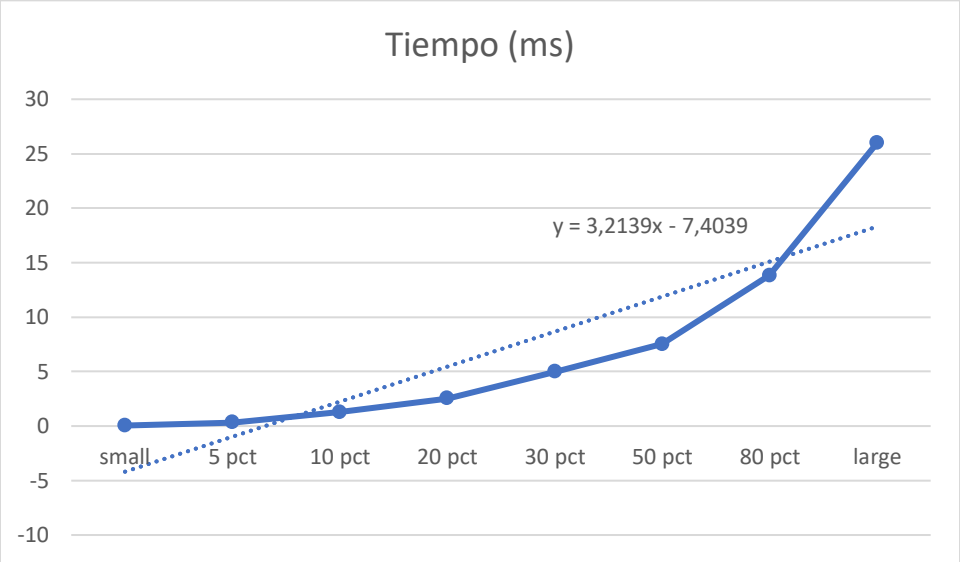
Procesadores	**AMD Ryzen 7 4800HS with Radeon Graphics**

Memoria RAM	8 GB
Sistema Operativo	Windows 10

Entrada	**Tiempo (ms)**
small	0.05
5 pct	0.33
10 pct	1.28
20 pct	2.54
30 pct	4.98
50 pct	7.51
80 pct	13.81
large	25.97

Graficas

Las gráficas con la representación de las pruebas realizadas.



Análisis

A pesar de que obtener un elemento en un *ArrayList,* dada su posición, tiene complejidad constante, la implementación de este requerimiento tiene un orden lineal O(n). Esto debido a que, lo primero que se hace es verificar si el elemento hace parte de la lista. Específicamente, a la hora de buscar un elemento en una lista, en el peor de los casos es necesario recorrer toda la lista, es decir, complejidad 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.