Universidad de los Andes
Facultad de Ingeniería

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

ABET Engineering Accreditation Commission

ANÁLISIS DEL RETO

Juan Pablo Castro, 202222086, jp.castrob1@uniandes.edu.co

Marcos Rodrigo España, 202124714, m.espanac@uniandes.edu.co

Gabriela Escobar, 201130766, g.escobar23@uniandes.edu.co

Requerimiento <<1>>

Plantilla para el documentar y analizar cada uno de los requerimientos.

Descripción

Breve descripción de como abordaron la implementación del requerimiento

Entrada	No se tienen parámetros de entrada
Salidas	La actividad económica con mayor total saldo a pagar de todos los años
Implementado (Sí/No)	Sí se implementó. Este era un requerimiento grupal, lo realizaron Marcos y Gabriela.

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

ASIGNACIONES	**Complejidad**
Final mayor, revisa en un FOR los años de a_poranios y luego, en otro FOR, todos los elementos de mayor[anio].	O(10 * 11)
`for i in lt.iterator(data_structs["data"])` \ `for Anios in a_poranios.keys():`	O(N) ~2N
`merg.sort(a_poranios[Anios],CMP_ValActEcon)`	O($_N log_N$)
TOTAL/ peor caso =	***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	11th Gen Intel(R) Core(TM) i7-1195G7 @ 2.90GHz 2.92 GHz
Memoria RAM	32GB
Sistema Operativo	Windows 11 Pro- 64 bits

Entrada	Tiempo (ms)
small	25
5	28
10	35
20	30
30	56
50	85
80	90
Large	110

Tablas de datos

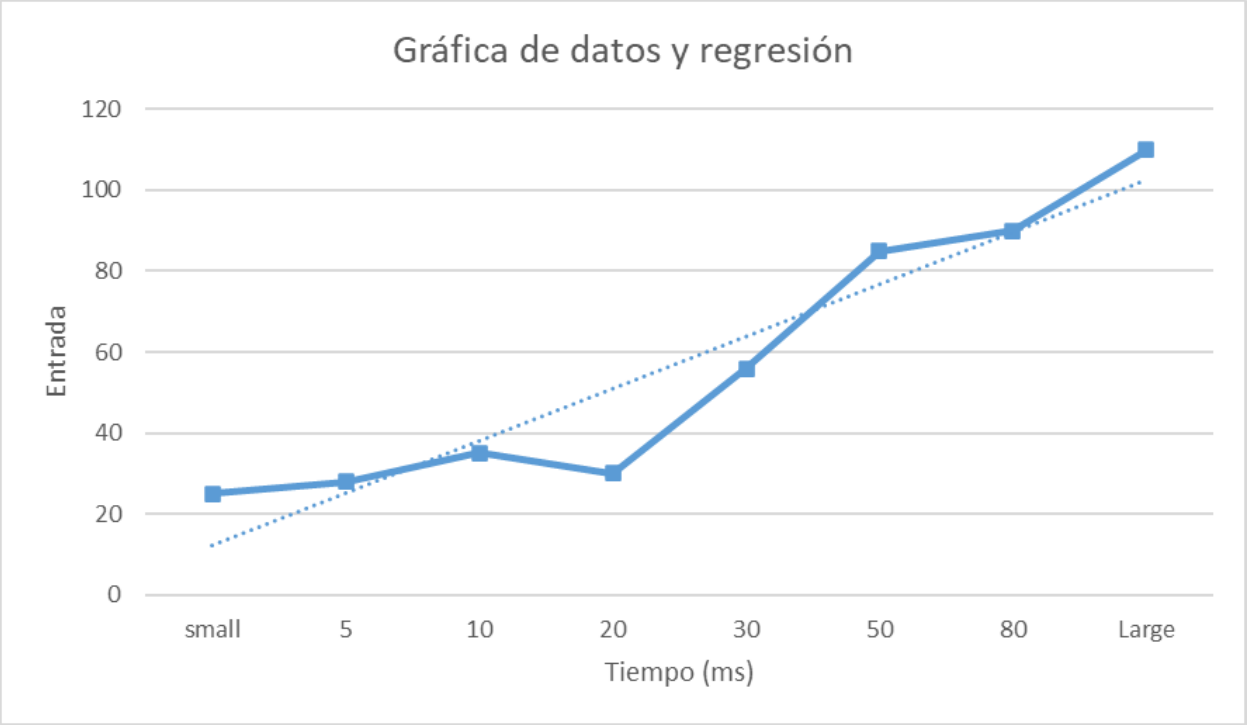
Las tablas con la recopilación de datos de las pruebas.

Muestra	Salida	Tiempo (ms)
small	Dato1	25
5 pct	Dato2	28
10 pct	Dato3	35
20 pct	Dato4	30
30 pct	Dato5	56
50 pct	Dato6	85
80 pct	Dato7	90

large	Dato8	110

Graficas

Las gráficas con la representación de las pruebas realizadas.



Análisis

Para poder hacer el siguiente análisis, cabe resaltar que en la complejidad temporal, N se refiere a un recorrido en todo el archivo. En este caso, N sería un recorrido en todas las actividades económicas. En primer lugar, con respecto a las complejidades temporales, al tener un for dentro de otro, usualmente tendría O(N^2). Sin embargo, este es un caso especial, pues la primera iteración se hace sobre la cantidad de años analizados (10), y la segunda se hace sobre las características analizadas de las mayores actividades económicas (11) de los años. Estas iteraciones nunca puede llegar a ser ni igual ni mayor que N, son de recorridos parciales. Con respecto a la gráfica y a las pruebas, se puede corroborar que este requerimiento tuvo una complejidad temporal lineal.

Requerimiento <<2>>

Plantilla para el documentar y analizar cada uno de los requerimientos.

Descripción

Breve descripción de como abordaron la implementación del requerimiento

Entrada	No se tienen parámetros de entrada
Salidas	La actividad económica con mayor total saldo a favor de todos los años
Implementado (Sí/No)	Sí se implemento. Este era un requerimiento grupal, lo realizaron Marcos y Gabriela.

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
Final mayor, revisa en un FOR los Total_Saldo_Favor de a_poranios y luego, en otro FOR, todos los elementos de mayor[Total_Saldo_Favor].	O(10*10)
For por todos los datos	O(N)
Paso ….	O(...)
TOTAL/ peor de los casos	***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).

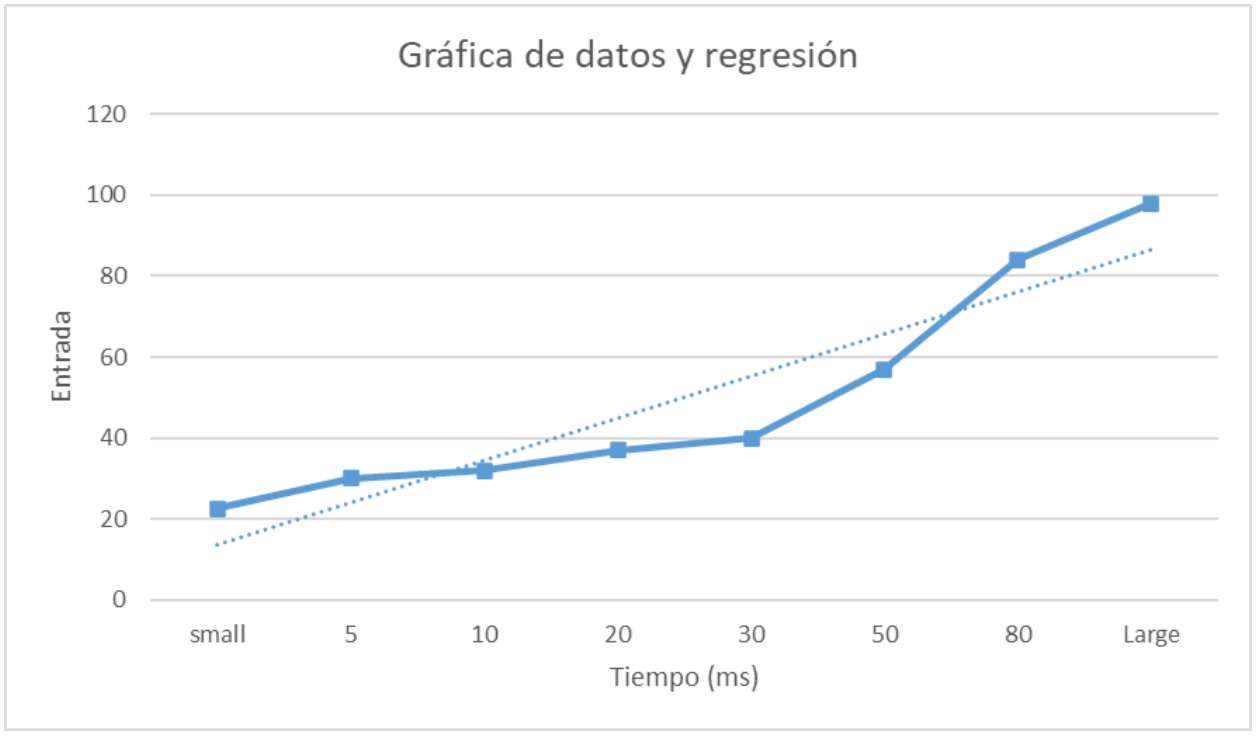
Entrada	**Tiempo (ms)**
small	22,5
5	30
10	32
20	37
30	40
50	57
80	84
Large	98

Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

Muestra	Salida	Tiempo (ms)
small	Dato1	22,5
5 pct	Dato2	30
10 pct	Dato3	32
20 pct	Dato4	37
30 pct	Dato5	40
50 pct	Dato6	57
80 pct	Dato7	84
large	Dato8	98

Gráficas



Análisis

Este requerimiento era similar al anterior, y esto se puede comprobar por la complejidad y la gráfica. Tiene una complejidad lineal, porque si se analiza el peor de los casos, se tendrían que revisar todos los elementos del archivo.

Requerimiento <<3>>

Plantilla para el documentar y analizar cada uno de los requerimientos.

Descripción

Breve descripción de como abordaron la implementación del requerimiento

Entrada	No se tienen parámetros de entrada
Salidas	Subsector económico con el menor total de retenciones para todos los años. Tres actividades económicas que más y menos aportaron en cada año.
Implementado (Sí/No)	Sí se implementó. Este era un requerimiento individual, lo realizó Juan Pablo

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
For que recorre todos los datos de un diccionario organizado por años, luego for que recorre una copia del diccionario inicial, luego un for por cada elemento de esta copia	$O(N^3)$
TOTAL	$\boldsymbol{O(N^3)}$

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).

Entrada	Tiempo (ms)
small	78
5	98
10	458,9

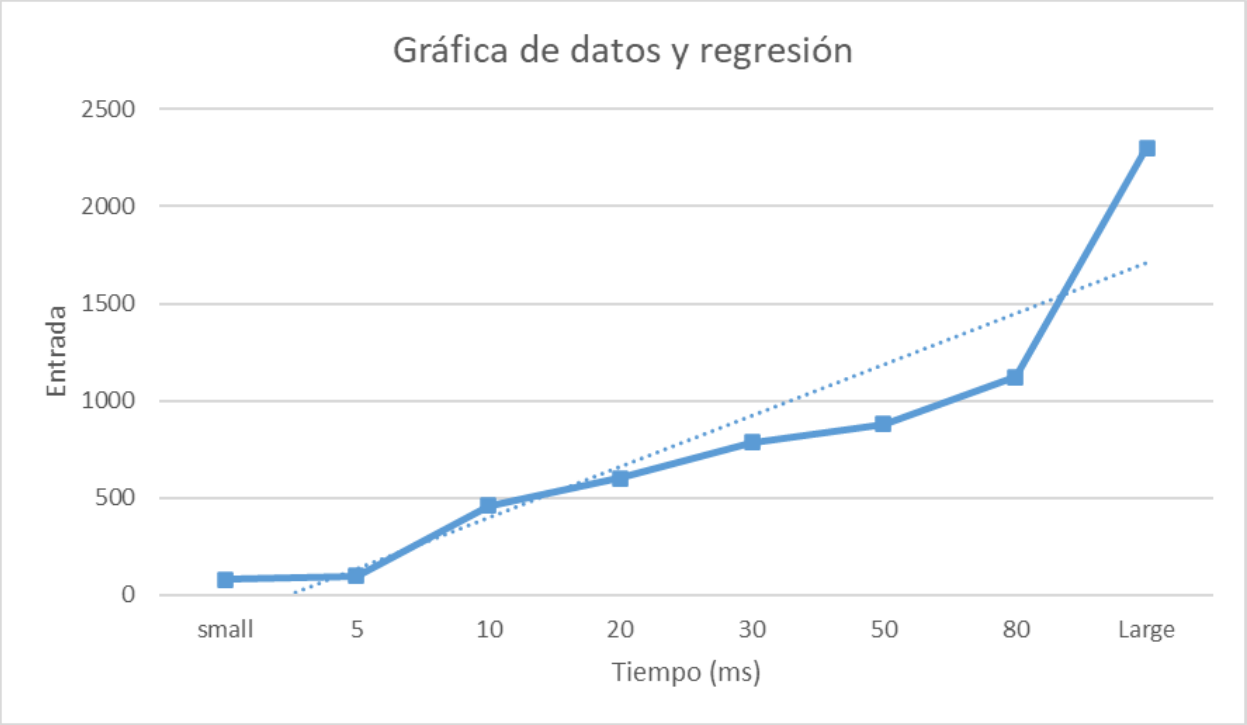
20	600
30	786,9
50	879,89
80	1123,45
Large	2300,78

Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

Muestra	Salida	Tiempo (ms)
small	Dato1	78
5 pct	Dato2	98
10 pct	Dato3	458,9
20 pct	Dato4	600
30 pct	Dato5	786,9
50 pct	Dato6	879,89
80 pct	Dato7	1123,45
large	Dato8	2300,78

Graficas



Análisis

Aunque la gráfica no es exactamente como debería ser en teoría un algoritmo con complejidad al cubo, se acerca mucho a esta realidad. Con respecto al orden de complejidad, un ordenamiento con complejidad cúbica es muy poco eficiente. Sin embargo, se puede afirmar, que tomando en cuenta nuestro conocimiento actual de TADS y la complejidad de este requerimiento, se realizó lo que se consideraba adecuado. En un futuro cercano, se va a poder realizar correctamente el requerimiento, con una complejidad temporal menor.

Requerimiento <<4>>

Recibe la data, despues la separa por años y hacer

Este req recibe un diccionario con la data separada por años ya, hace una copia de la data y la recorre para luego llamar a un método que vuelve y separa por cada subsector la data ya separada por año.

Suma el total de descuentos tributarios para cada subsector en un año y luego verifica si es mayor al anterior o no, si lo es lo cambia y así por toda la data. Después en final1 retornamos en un arraylist la lista final de el que tuvo el mayor descuento tributario para cada uno de los años. Despues hago un merge para organizarlos y mostrar la función y con las posiciones organizadas saco los 3 mayores y los 3 menores, lo guardo en final_act_econ y devuelvo eso al controller

Descripción

Breve descripción de como abordaron la implementación del requerimiento

Entrada	Recibe `data_structs que es la data`
Salidas	Final1 retorna la lista con los mayores `Costos y gastos nómina` por los años disponibles. Headers: Retorna los títulos de los encabezados del tabulate final_act_econ: retorna la lista final de actores económicos
Implementado (Sí/No)	Si se implementó , realizado por Gabriela

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	**Complejidad**
For que crea un diccionario organizando los datos por año. Otro for que creará la posibilidad de sumar los valores. Otro for que suma por subsector Costo y gasto por nómina.	$O(N^3)$
TOTAL	$O(N^3)$

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).

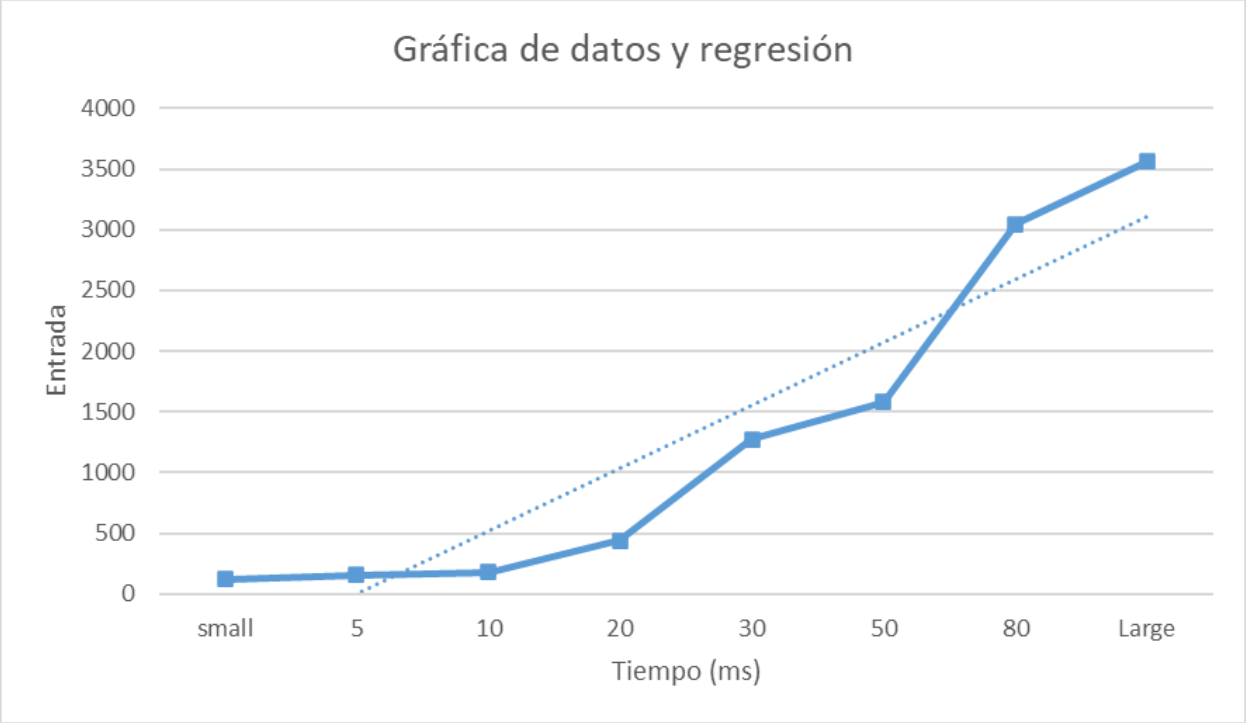
Entrada	Tiempo (ms)
small	123,4
5	156,7
10	178,9
20	437,6
30	1274,9
50	1583,2
80	3048,5
Large	3562,8

Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

Muestra	Salida	Tiempo (ms)
small	Dato1	123,4
5 pct	Dato2	156,7
10 pct	Dato3	178,9
20 pct	Dato4	437,6
30 pct	Dato5	1274,9
50 pct	Dato6	1583,2
80 pct	Dato7	3048,5
large	Dato8	3562,8

Gráficas



Análisis

Con similitud al problema anterior, la complejidad de este requerimiento no es eficiente. Podría haber soluciones más eficientes pero considerando nuestro conocimiento actual de estructuras de datos, esto fue lo que se consideró más apropiado para completar el requerimiento. La gráfica está acorde a la complejidad temporal.

Requerimiento <<5>>

Requerimiento 5

Este req recibe un diccionario con la data separada por años ya, hace una copia de la data y la recorre para luego llamar a un método que vuelve y separa por cada subsector la data ya separada por año.

Suma el total de descuentos tributarios para cada subsector en un año y luego verifica si es mayor al anterior o no, si lo es lo cambia y así por toda la data. Después en final1 retornamos en un arraylist la lista final de el que tuvo el mayor descuento tributario para cada uno de los años. Despues hago un merge para organizarlos y mostrar la función y con las posiciones organizadas saco los 3 mayores y los 3 menores, lo guardo en final_act_econ y devuelvo eso al controller

Descripción

Entrada	Entra la base de datos ya antes separado por años en un diccionario
Salidas	F1 retorna la lista con los mayores descuentos tributarios por los años disponibles. Headers: Retorna los títulos de los encabezados del tabulate factecon: retorna la lista final de actores económicos
Implementado (Sí/No)	Si se implemento, realizado por Marcos

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	Complejidad
`for anio in por_anios.keys():`	O(n)
`temp[anio] = organizar_por_subsector(temp[anio])`	O(n) esta es la complejidad del método llamado
`for subsector in temp[anio].keys():`	O(n)
`for i in temp[anio][subsector]["elements"]:`	O(n)
`if total_dtributarios > menor:`	O(1)

`for i in temp[anio][sub_menor]["elements"]:`	O(n)
`sub_temp = lt.getElement(temp[anio][sub_menor], 1)`	O(1) En un array un Get es o(1), en un single es O(N)
`lt.addLast(final1, temp1)`	O(1)
`merg.sort(por_anios[anio], cmp_dtribu)`	O(nlogn)
`size = lt.size(por_anios[anio])`	O(1)
`for i in lt.iterator(data_structs["data"]):`	O(n)
`if i ['Año'] not in a_poranios:`	O(1)
TOTAL	***O(***n^3***)***

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).

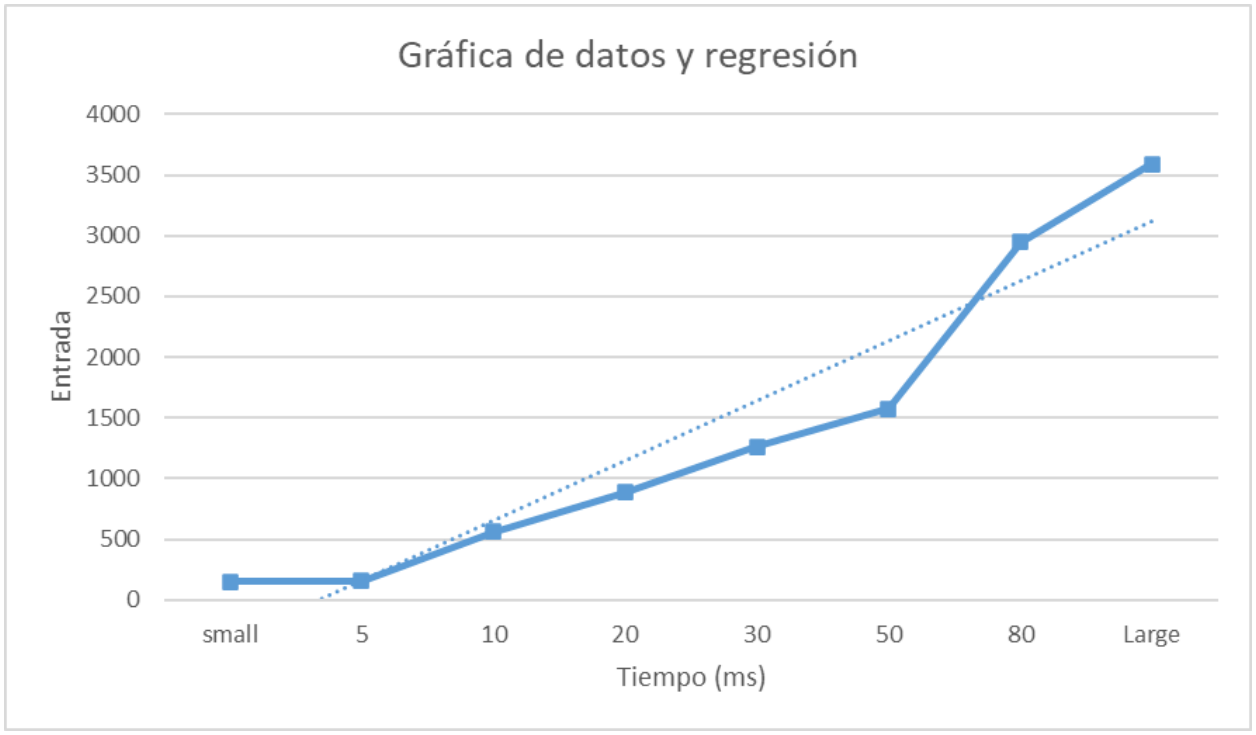
Entrada	Tiempo (ms)
small	147,9
5	156,7
10	563,95
20	885,34
30	1262,7
50	1575,6
80	2948,5
Large	3592,8

Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

Muestra	Salida	Tiempo (ms)
small	Dato1	147,9
5 pct	Dato2	156,7
10 pct	Dato3	563,95
20 pct	Dato4	885,34
30 pct	Dato5	1262,7
50 pct	Dato6	1575,6
80 pct	Dato7	2948,5
large	Dato8	3592,8

Gráficas



Análisis

La complejidad de este algoritmo viene siendo muy alta ya que es muy ineficiente y no permite que el algoritmo se desarrolle de una manera rápida, el ejemplo más claro está en los bucles for los cuales elevan la complejidad hasta un N^3. Mediante diferentes estructuras se podría agilizar este algoritmo la implementación de estos todavía no es muy clara

Requerimiento <<6>>

Plantilla para el documentar y analizar cada uno de los requerimientos.

Descripción

Breve descripción de como abordaron la implementación del requerimiento

Entrada	Debe recibir el año como parámetro de entrada
Salidas	La actividad económica con mayor total de ingresos netos para cada sector económico en un año específico.
Implementado (Sí/No)	Sí se implementó. Este era un requerimiento grupal y lo desarrolló, casi en su totalidad, Juan Pablo.

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	Complejidad
For, que recorre los elementos del diccionario elem_anio. Otro For, en el que se recorren los elementos del diccionario elem_anio[sector]. Otro for, en el que se recorre el subdiccionario elem_anio[sector][subsector]	$O(N^3)$
Dentro de cada for hay operaciones constantes	O(1)
TOTAL	$\boldsymbol{O(N^3)}$

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).

Entrada	Tiempo (ms)
small	147,9
5	156,7

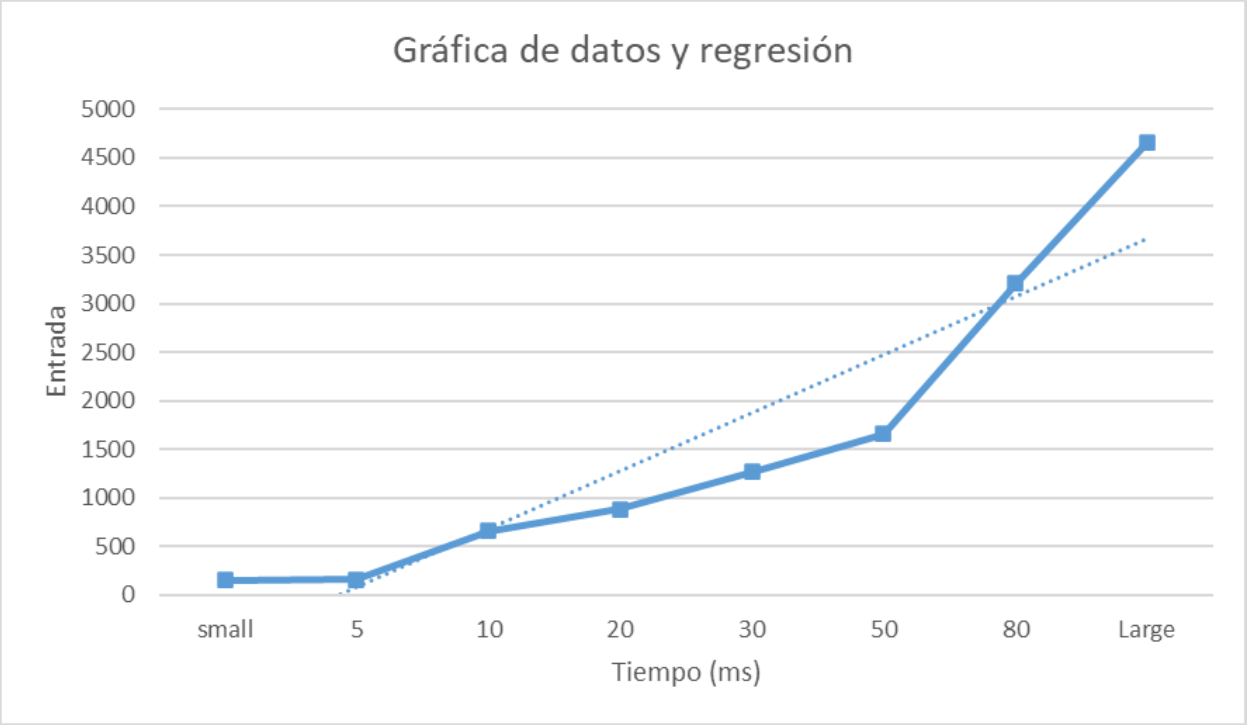
10	654,8
20	885,34
30	1262,7
50	1654,3
80	3207,2
Large	4656,8

Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

Muestra	Salida	Tiempo (ms)
small	Dato1	147,9
5 pct	Dato2	156,7
10 pct	Dato3	654,8
20 pct	Dato4	885,34
30 pct	Dato5	1262,7
50 pct	Dato6	1654,3
80 pct	Dato7	3207,2
large	Dato8	4656,8

Graficas



Análisis

Se pudo llegar a esta complejidad, gracias al siguiente raciocinio: dado que el primer ciclo recorre los elementos del diccionario "elem_anio", el segundo ciclo recorre los elementos del diccionario "elem_anio[sector]" y el tercer ciclo recorre los elementos del subdiccionario "elem_anio[sector][subsector]". Dentro de cada uno de estos ciclos hay una serie de operaciones que se realizan en tiempo constante. Por lo tanto, la complejidad temporal total del algoritmo depende del número de elementos en cada nivel del diccionario. Si tenemos n elementos en el primer nivel, m elementos en el segundo nivel y k elementos en el tercer nivel, la complejidad sería de O(n*m*k). Sin embargo, como no sabemos las dimensiones de los diccionarios que se están utilizando en este algoritmo, lo más prudente es referirnos a la complejidad temporal como O(n^3). Tomando en cuenta que este requerimiento era de nivel avanzado, utilizando listas TAD, la complejidad cúbica corresponde lo máximo que nuestras habilidades actuales podían alcanzar. A diferencia de los requerimientos anteriores, en este caso la complejidad va muy acorde a los resultados de las pruebas.

Requerimiento <<7>>

Plantilla para el documentar y analizar cada uno de los requerimientos.

Descripción

Breve descripción de como abordaron la implementación del requerimiento

Entrada	Data structs, top: número de actividades a identificar, año inicial,año final
Salidas	Final: Es la lista con las actividades identificadas dentro de ese periodo de tiempo
Implementado (Sí/No)	Sí se implemento. Juan Pablo

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	Complejidad

`for datos in lt.iterator(data_structs["data"]):`
`  if int(datos["Año"]) in range(int(anio1), int(anio2)+1):`
`  lt.addLast(datos_en_rango, datos)`
	O(n)

`sa.sort(datos_en_rango, cmp_tcyg)`	$O(N^3)$
`if int(top)> lt.size(datos_en_rango):` `  rango = range(1, lt.size(datos_en_rango))`	O(1)

`for n in rango:`
`  x = lt.getElement(datos_en_rango, n)`
`  temp1 = {}`
`  temp1["Año"] = x["Año"]`
`  temp1["Código actividad económica"] = x["Código actividad económica"]`
`  temp1["Nombre actividad económica"] = x["Nombre actividad económica"]`
`  temp1["Código sector económico"] = x["Código sector económico"]`
`  temp1["Nombre sector económico"] = x["Nombre sector económico"]`
`  temp1["Total ingresos netos consolidados para periodo"] = x["Total ingresos netos"]`
`  temp1["Total costos y gastos a pagar para periodo consolidado"] = x["Total costos y gast`
`  temp1["Total saldo a pagar para periodo consolidado"] = x["Total saldo a pagar"]`
`  temp1["Total saldo a favor para periodo consolidado"] = x["Total saldo a favor"]`
	O(n)

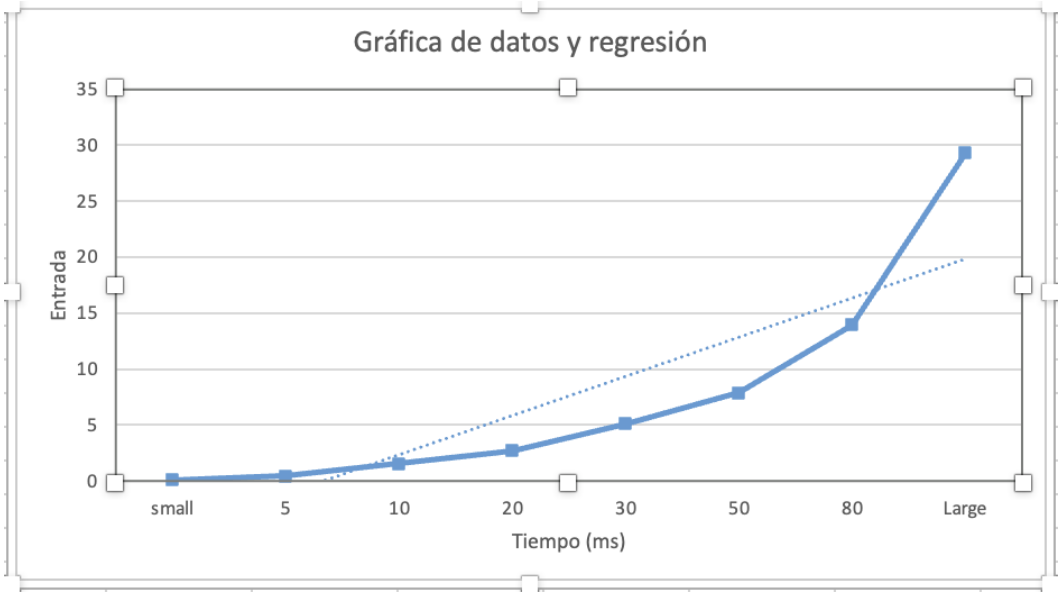
`lt.addLast(final, temp1)`	O(1)
TOTAL	$O(N^3)$

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).

Tablas de datos

Las tablas con la recopilación de datos de las pruebas.



Graficas

Las gráficas con la representación de las pruebas realizadas.

Entrada	Tiempo (ms)
small	0,08
5	0,44
10	1,56
20	2,7
30	5,1
50	7,86
80	13,93
Large	29,34

Análisis

Este algoritmo es uno de los más rápidos pero lo único que logró afectar su complejidad es el uso del sort que le elevo su complejidad al cubo, pero si no hubiera sido este su complejidad hubiera sido de N ya que no tiene ningún for anidado

Requerimiento <<8>>

Plantilla para documentar y analizar cada uno de los requerimientos.

Descripción

Como analista económico **necesito** identificar las **N** actividades económicas de todos y cada uno de los subsectores económicos que tuvieron los mayores valores totales de impuestos a cargo (**Total Impuesto a cargo**), para un periodo de tiempo específico.

En este req se busca las actividades de los sub económicos con los mayores valores totales de impuestos a cargo para un periodo de tiempo específico, así que se recorre la data entre ese rango y se la agrega en una arraylist, para después cuando se agregue toda la del periodo de tiempo se le hace un sort para organizarlo con respecto al total de impuesto, después se define el rango y se saca en una lista array se saca el rango

Entrada	`data_structs: aca va la data , top: numero de elementos que desea ver, anio1: año de inicio:, anio2: año de fin`
Salidas	`final: la lista con toda la info de los N en el periodo definidio`
Implementado (Sí/No)	Sí se implemento. Este era un requerimiento grupal, lo realizaron Marcos y Gabriela.

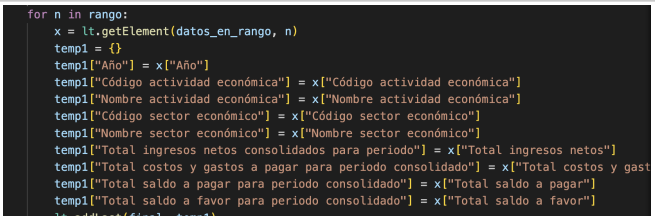
Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

Pasos	Complejidad

`datos_en_rango = lt.newList("ARRAY_LIST")`
`for datos in lt.iterator(data_structs["data"]):`
`  if int(datos["Año"]) in range(int(anio1), int(anio2)+1):`
`  lt.addLast(datos_en_rango, datos)`
	O(n)

`sa.sort(datos_en_rango, cmp_tic)`	O(nlogn)

	O(n)
	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).

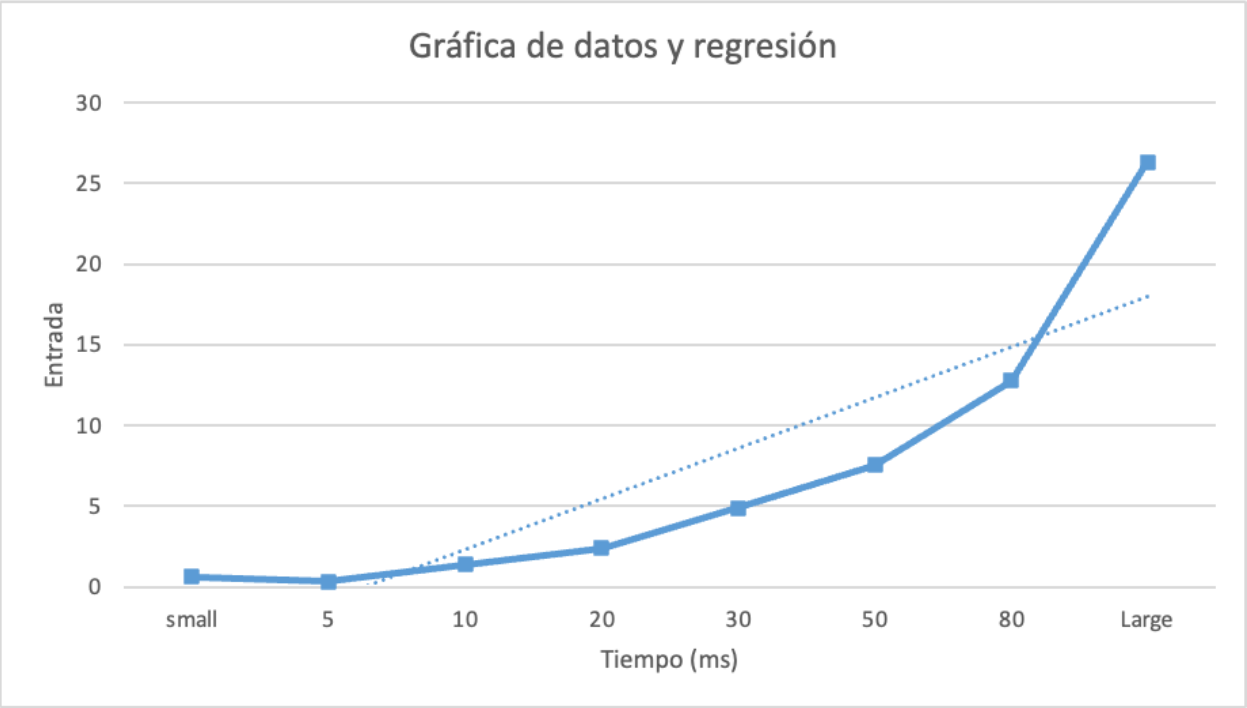
Entrada	Tiempo (ms)
small	0,6
5	0,33
10	1,37
20	2,4
30	4,9
50	7,53
80	12,78
Large	26,34

Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

Graficas

Las gráficas con la representación de las pruebas realizadas.



Análisis

Este es el segundo algoritmo con mejor complejidad aunque, siendo afectada por el sort, fue un algoritmo relativamente fácil de realizar ya que con todo lo hecho anteriormente la idea del algoritmo era casi lo mismo solo que con un criterio diferente.