ANÁLISIS DEL RETO 2

Simón Calderón López, 202113559, s.calderon@uniandes.edu.co

Alejandro Torres Rodríguez, 202013743, a.torres48@uniandes.edu.co

Johannes Almas, 202224983, j.almas@uniandes.edu.co

La estructura de datos usada para este reto fue la siguiente:

Todas las listas son tipo “ARRAY_LIST”, todos los mapas son tipo “PROBING” y el ordenamiento que usamos fue “Merge sort”.

Tenemos un objeto llamado DataStructs que tiene como atributos:

- all_data: que es una lista con todas las actividades económicas.
- map_by_year: que es un mapa por años de los registros.
- map_by_subsectors: que es un mapa por subsectores de ese año

Luego tenemos un objeto llamado Year que recibe al objeto DataStructs, Year tiene los atributos:

- all_data: que es una lista con todas las actividades económicas de ese año.
- map_by_sector: que es un mapa por sectores de ese año.
- subsector_max: aquí se guarda el subsector máximo de acuerdo con el sort_criteria.
- subsector_min: aquí se guarda el subsector mínimo de acuerdo con el sort_criteria.

Year tiene los siguientes métodos:

- search_min_max_subsector: este método encuentra el máximo y mínimo subsector.
- Search_min_max_sector: este método encuentra el máximo y mínimo sector.

Luego del objeto Year está el objeto Sector que recibe a Year, y tiene los siguientes atributos:

- all_data: que es una lista con todas las actividades económicas de ese sector.
- map_by_subsector: que es un mapa por subsectores de ese sector.
- max_total_payable_balance: atributo para el requerimiento 1
- min_total_payable balance: atributo para el requerimiento 1
- max_total_favorable_balance: atributo para el requerimiento 2
- min_total_favorable_balance: atributo para el requerimiento 2
- Dict_data: atributo para el requerimiento 3, 4 y 5.
- Total_all_net_incomes: atributo para el requerimiento 3, 4 y 5.
- Total_all_costs_and_expenses: atributo para el requerimiento 3, 4 y 5.
- Total_all_payable_balance: atributo para el requerimiento 3, 4 y 5.
- Total_all_favorable_balance: atributo para el requerimiento 3, 4 y 5.
- Less_apport_subsector: atributo para el requerimiento 3, 4 y 5.
- More_apport_subsector: atributo para el requerimiento 3, 4 y 5.
- Name_sector: atributos propios del sector.

- Code_sector: atributos propios del sector.

Sector tiene los siguientes métodos:

- Give_attributes
- Actualize
- Actualize_dict
- Sum_values
- Obtain_max_and_min_economic_activity
- Obtian_max_and_min_subsector
- Create_table_sector
- _reformatColumns
- Create_list
- _match_columns

Después de sector está el objeto Subsector que tiene los siguientes atributos:

- All_data: que es una lista de todas las actividades ecoónmicas por subsector.
- Total_retencions: atributo para el requerimiento 3.
- Total_costs_and_payroll_expenses: atributo para el requerimiento 4.
- Tax_discounts: atributo para el requerimiento 5.
- Total_net_incomes: atributo para el requerimiento 6.
- Total_costs_and_expenses: atributo para el requerimiento 7.

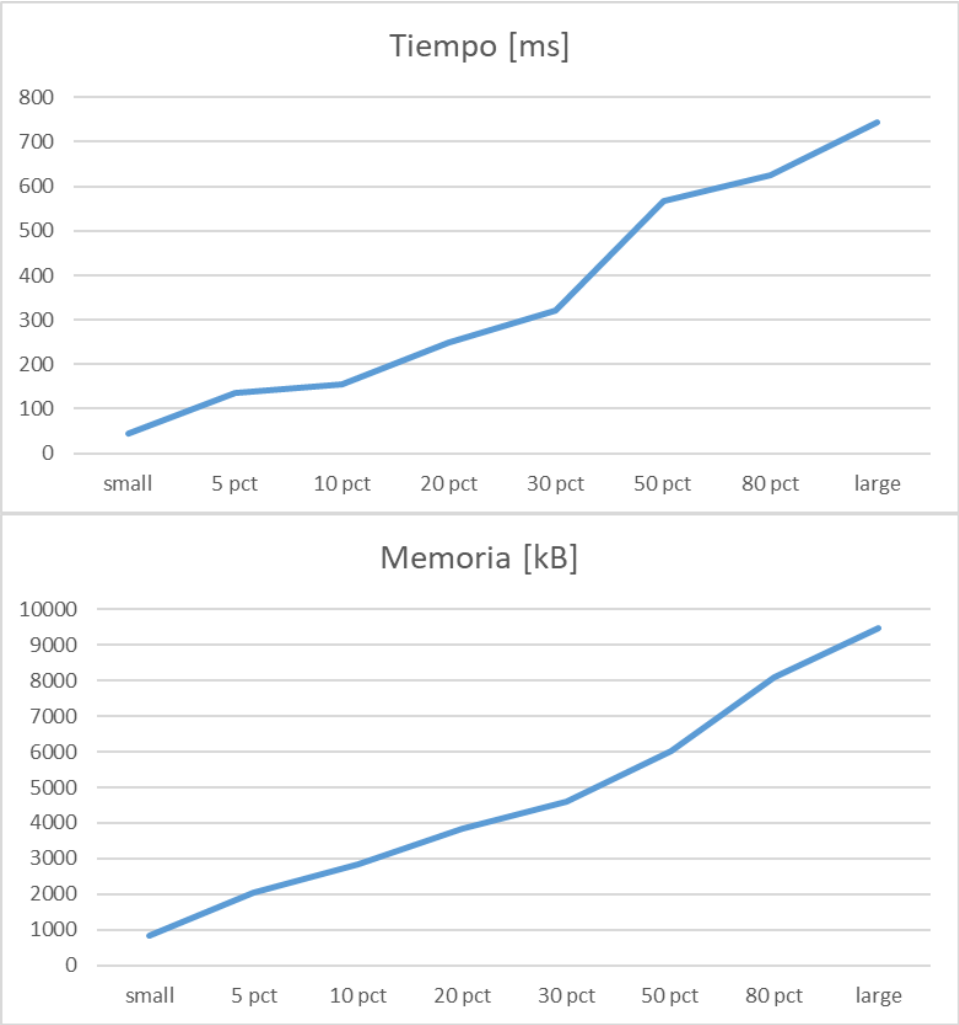
Pruebas Realizadas

Procesadores	**Intel(R) Core(TM) i5-9300H @2.40Ghz**
Memoria RAM	8 GB
Sistema Operativo	Windows 11 Home

Tabla de datos Carga

Carga	Tiempo (ms)	Memoria (kB)
small	45.957	843.143
5pct	135.911	2041.479
10pct	154.024	2846.766
20pct	249.011	3829.736
30pct	321.361	4614.474
50pct	566.726	6023.896
80pct	626.285	8086.61
large	743.468	9458.194

Graficas



Requerimiento 1

Obtener la actividad económica con mayor saldo a pagar para un sector económico y un año específico.

Descripción

Entrada	Sector económico, año.
Salidas	Actividad económica con mayor saldo a pagar.
Implementado (Sí/No)	Sí/ El grupo.

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo. Hay varios pasos que son O(1) que saltamos.

Pasos	**Complejidad**

Buscar la actividad económica. (En model.)	O(k*log(k)), k actividades en el sector. (Un merge sort).
TOTAL	***O(k*log(k))***

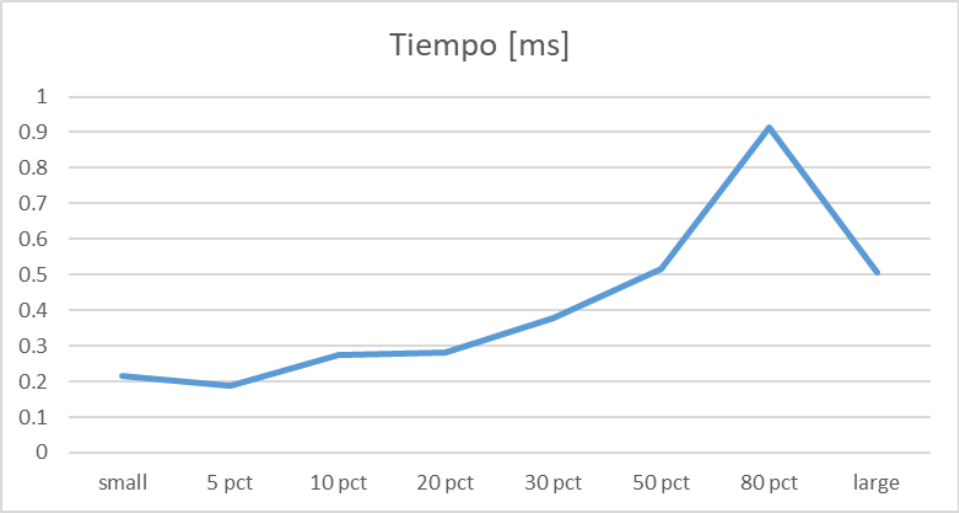
Pruebas Realizadas

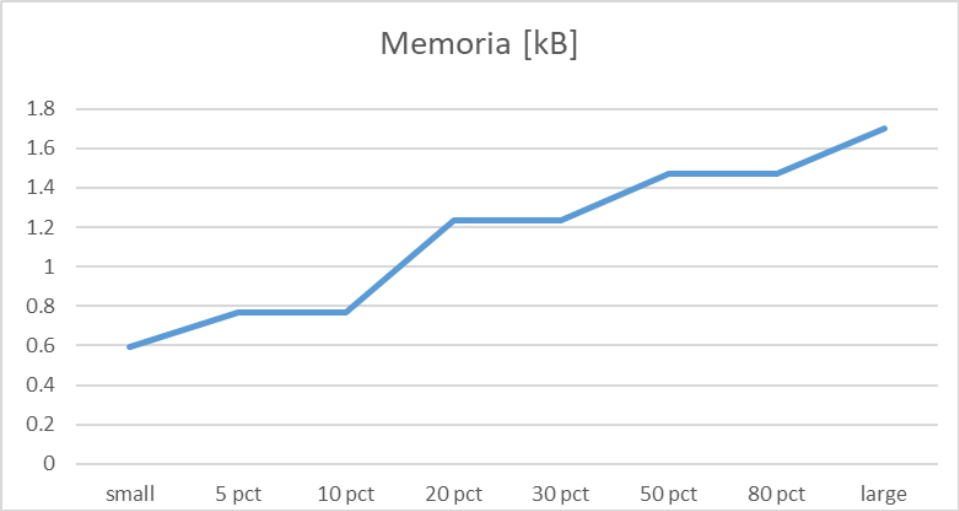
Procesadores	**Intel(R) Core(TM) i5-9300H @2.40Ghz**
Memoria RAM	8 GB
Sistema Operativo	Windows 11 Home

Tablas de datos

Req 1	Tiempo (ms)	Memoria (kB)
small	0.216	0.594
5pct	0.189	0.766
10pct	0.273	0.766
20pct	0.281	1.234
30pct	0.378	1.234
50pct	0.516	1.469
80pct	0.915	1.469
large	0.507	1.703

Graficas





Análisis

El algoritmo tiene una complejidad temporal de O(k*log(k)), si hay k actividades económicas dentro del sector. Esto es porque el algoritmo solo tiene pasos de complejidad constante, salvo el paso de ordenar las actividades.

Requerimiento 2

Obtener la actividad económica con mayor saldo a favor para un sector económico y un año específico.

Descripción

Entrada	Sector económico, año
Salidas	Actividad económica con mayor saldo a favor.
Implementado (Sí/No)	Si/ El grupo.

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo. Hay varios pasos que son O(1) que saltamos.

Pasos	**Complejidad**
Buscar la actividad económica. (En model.)	O(k*log(k)), k actividades en el sector. (Un merge sort).
TOTAL	***O(k*log(k))***

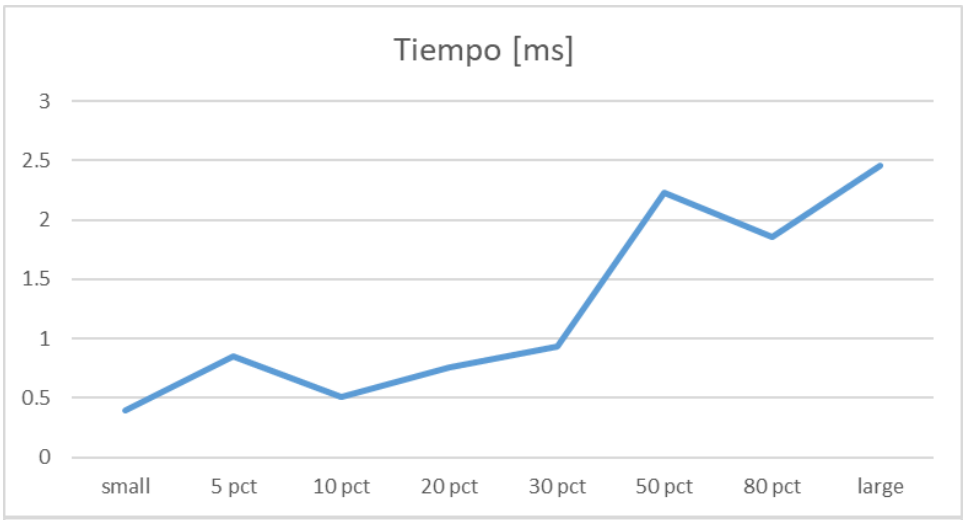
Pruebas Realizadas

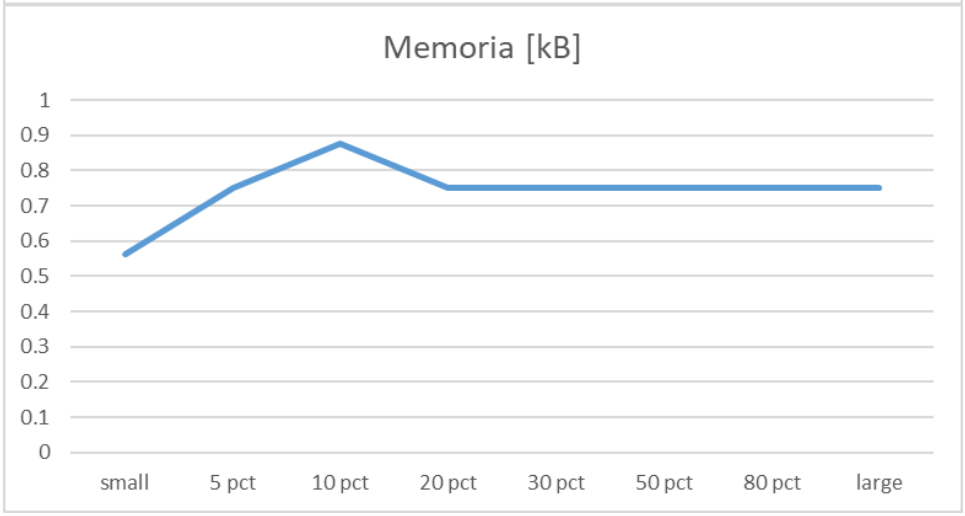
Procesadores	**Intel(R) Core(TM) i5-9300H @2.40Ghz**
Memoria RAM	8 GB
Sistema Operativo	Windows 11 Home

Tablas de datos

Req 2	Tiempo (ms)	Memoria (kB)
small	0.399	0.562
5pct	0.85	0.75
10pct	0.508	0.875
20pct	0.753	0.75
30pct	0.934	0.75
50pct	2.234	0.75
80pct	1.851	0.75
large	2.454	0.75

Graficas





Análisis

El algoritmo tiene una complejidad temporal de O(k*log(k)), si hay k actividades económicas dentro del sector. Esto es porque el algoritmo solo tiene pasos de complejidad constante, salvo el paso de ordenar las actividades.

Requerimiento 3

Encontrar el subsector económico con el menor total de retenciones para un año especifico.

Descripción

Entrada	Año.
Salidas	Subsector económico con el menor total de rentenciones.
Implementado (Sí/No)	Sí/ Simón Calderón

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo. Hay varios pasos que son O(1) que saltamos.

Pasos	**Complejidad**
Buscar el subsector. (En model.)	O(x*y*log(y)), x sectores, y subsectores. (Un merge sort x veces).
Ordenar las actividades económicas del subsector. (En model.)	O(k*logk), k actividades (Un merge sort).
Crear una tabla del subsector. (En view.)	O(K), K constante
Crear una tabla de las actividades económicas. (En view.)	O(K), K constante
TOTAL	***O(x*y*log(y))***

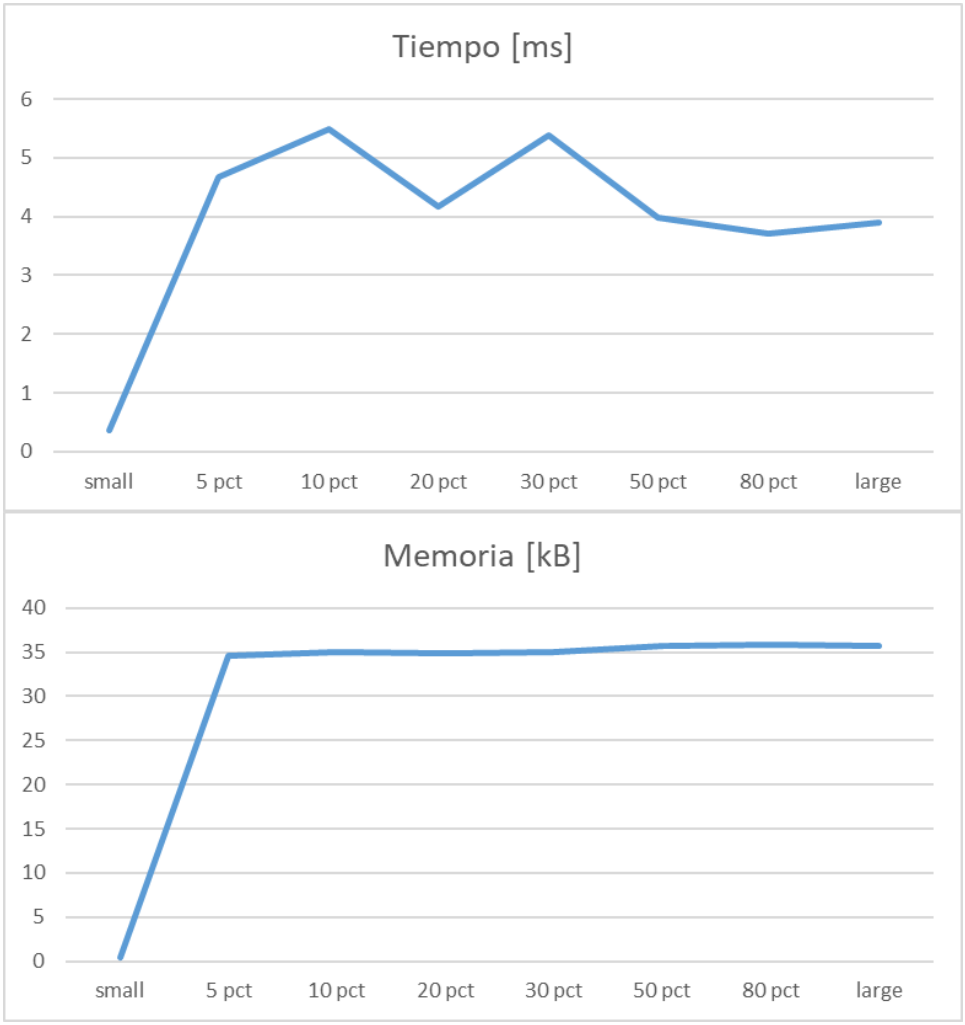
Pruebas Realizadas

Procesadores	**Intel(R) Core(TM) i5-9300H @2.40Ghz**
Memoria RAM	8 GB
Sistema Operativo	Windows 11 Home

Tablas de datos

Req 3	Tiempo (ms)	Memoria (kB)
small	0.353	0.422
5pct	4.666	34.639
10pct	5.491	34.951
20pct	4.163	34.889
30pct	5.392	34.951
50pct	3.973	35.732
80pct	3.719	35.857
large	3.894	35.732

Graficas



Análisis

El algoritmo tiene una complejidad temporal de O(x*y*log(y)), si hay x sectores y y subsectores. La cantidad de subsectores en un sector puede variar, pero definiendo y como el peor caso, nos da esta complejidad. La complejidad se debe a que la búsqueda del subsector económico es el paso con la mayor complejidad temporal. Este paso recorre los x sectores, haciendo un merge sort cada vez. El merge sort tiene una complejidad de O(y*log(y)) y lo hacemos x veces. En el caso donde hay muy pocos sectores y subsectores, pero un montón de actividades económicas, el merge sort del subsector (que tiene una complejidad que depende de k el número de actividades económicas) puede tener una complejidad peor que la complejidad de la búsqueda. Si queremos incluir este caso, podemos decir que la complejidad del algoritmo es O(x*y*log(y) + k*log(k)). La complejidad más sencilla - O(x*y*log(y)) - es más relevante en nuestro caso.

Requerimiento 4

Encontrar el subsector económico con los mayores costos y gastos de nómina para un año específico.

Descripción

Entrada	Año.
Salidas	Subsector económico con los mayores costos y gastos de nómina.
Implementado (Sí/No)	Sí/ Alejandro Torres Rodríguez

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo. Hay varios pasos que son O(1) que saltamos.

Pasos	**Complejidad**
Buscar el subsector. (En model.)	O(x*y*log(y)), x sectores, y subsectores. (Un merge sort x veces).
Ordenar las actividades económicas del subsector. (En model.)	O(k*logk), k actividades (Un merge sort).
Crear una tabla del subsector. (En view.)	O(K), K constante
Crear una tabla de las actividades económicas. (En view.)	O(K), K constante
TOTAL	***O(x*y*log(y))***

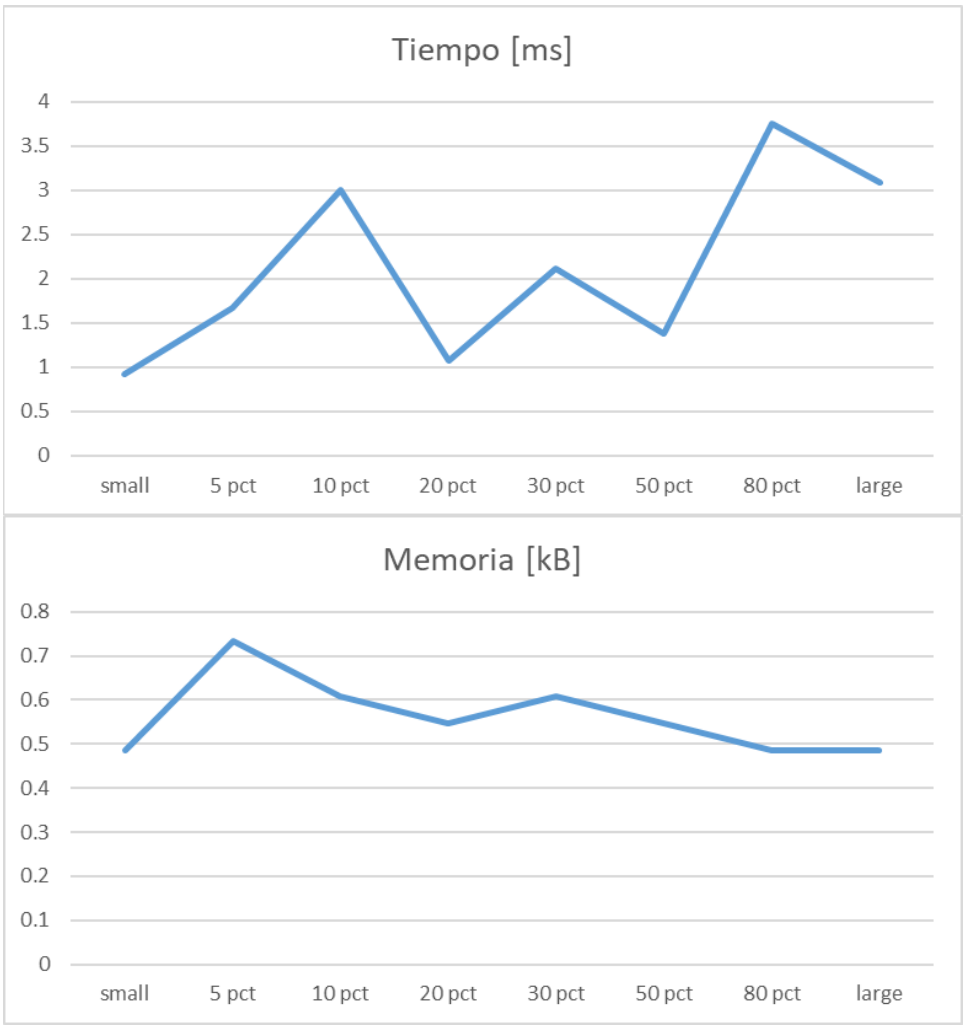
Pruebas Realizadas

Procesadores	**Intel(R) Core(TM) i5-9300H @2.40Ghz**
Memoria RAM	8 GB
Sistema Operativo	Windows 11 Home

Tablas de datos

Req 4	Tiempo (ms)	Memoria (kB)
small	0.917	0.484
5pct	1.677	0.734
10pct	3.005	0.609
20pct	1.068	0.547
30pct	2.114	0.609
50pct	1.38	0.547
80pct	3.747	0.484
large	3.092	0.484

Graficas



Análisis

El algoritmo tiene una complejidad temporal de O(x*y*log(y)), si hay x sectores y y subsectores. La cantidad de subsectores en un sector puede variar, pero definiendo y como el peor caso, nos da esta complejidad. La complejidad se debe a que la búsqueda del subsector económico es el paso con la mayor complejidad temporal. Este paso recorre los x sectores, haciendo un merge sort cada vez. El merge sort tiene una complejidad de O(y*log(y)) y lo hacemos x veces. En el caso donde hay muy pocos sectores y subsectores, pero un montón de actividades económicas, el merge sort del subsector (que tiene una complejidad que depende de k el número de actividades económicas) puede tener una complejidad peor que la complejidad de la búsqueda. Si queremos incluir este caso, podemos decir que la complejidad del algoritmo es O(x*y*log(y) + k*log(k)). La complejidad más sencilla - O(x*y*log(y)) - es más relevante en nuestro caso.

Requerimiento 5

Encontrar el subsector económico con los mayores descuentos tributarios para un año específico.

Descripción

Entrada	Año.
Salidas	Subsector económico con los mayores descuentos tributarios.
Implementado (Sí/No)	Sí/ Johannes Almas.

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo. Hay varios pasos que son O(1) que saltamos.

Pasos	**Complejidad**
Buscar el subsector. (En model.)	O(x*y*log(y)), x sectores, y subsectores. (Un merge sort x veces).
Ordenar las actividades económicas del subsector. (En model.)	O(k*log(k)), k actividades (Un merge sort).
Crear una tabla del subsector. (En view.)	O(K), K constante
Crear una tabla de las actividades económicas. (En view.)	O(K), K constante
TOTAL	***O(x*y*log(y))***

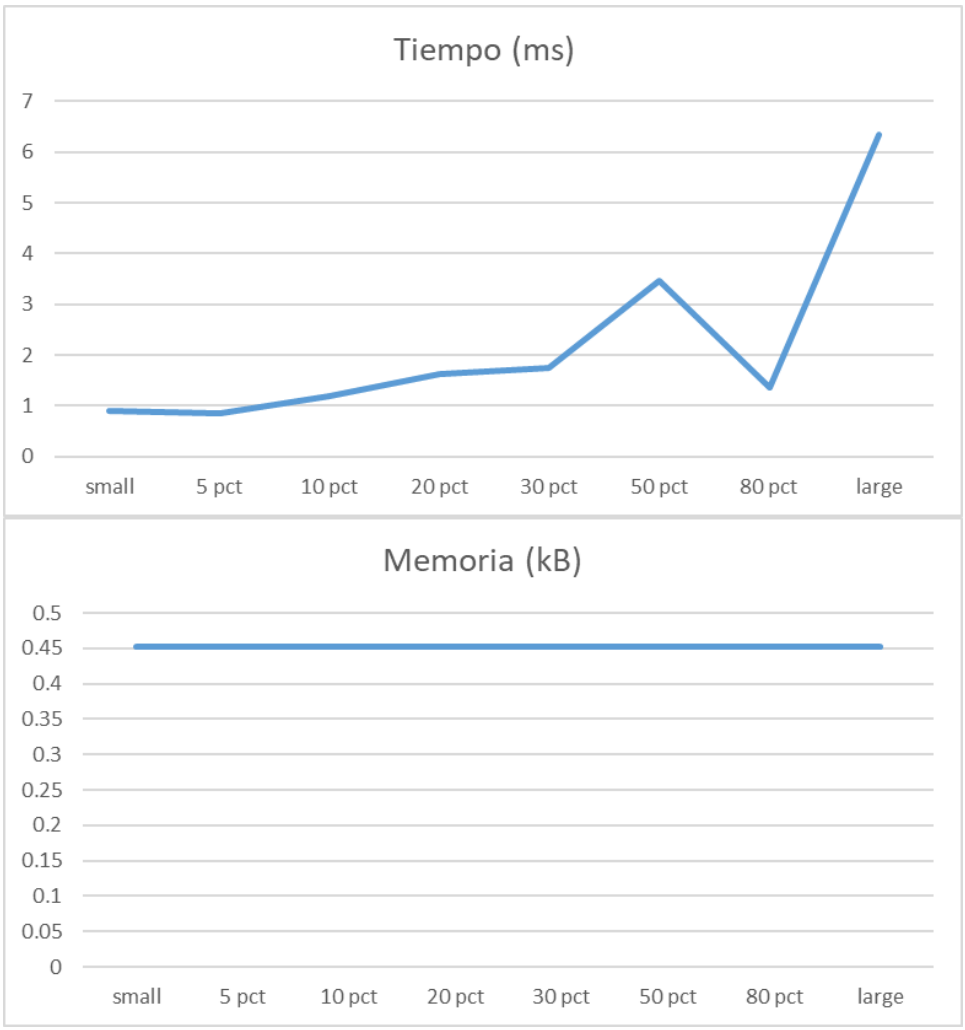
Pruebas Realizadas

Procesadores	**Intel(R) Core(TM) i5-9300H @2.40Ghz**
Memoria RAM	8 GB
Sistema Operativo	Windows 11 Home

Tablas de datos

Req 5	Tiempo (ms)	Memoria (kB)
small	0.892	0.453
5pct	0.846	0.453
10pct	1.182	0.453
20pct	1.635	0.453
30pct	1.756	0.453
50pct	3.545	0.453
80pct	1.362	0.453
large	6.337	0.453

Graficas



Análisis

El algoritmo tiene una complejidad temporal de O(x*y*log(y)), si hay x sectores y y subsectores. La cantidad de subsectores en un sector puede variar, pero definiendo y como el peor caso, nos da esta complejidad. La complejidad se debe a que la búsqueda del subsector económico es el paso con la mayor complejidad temporal. Este paso recorre los x sectores, haciendo un merge sort cada vez. El merge sort tiene una complejidad de O(y*log(y)) y lo hacemos x veces. En el caso donde hay muy pocos sectores y subsectores, pero un montón de actividades económicas, el merge sort del subsector (que tiene una complejidad que depende de k el número de actividades económicas) puede tener una complejidad peor que la complejidad de la búsqueda. Si queremos incluir este caso, podemos decir que la complejidad del algoritmo es O(x*y*log(y) + k*log(k)). La complejidad más sencilla - O(x*y*log(y)) - es más relevante en nuestro caso.

Requerimiento 6

Encontrar el sector económico con el mayor total de ingresos netos para un año específico.

Descripción

Entrada	Año.
Salidas	Sector económico con el mayor total de ingresos netos.
Implementado (Sí/No)	Sí/ El grupo.

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo. Hay varios pasos que son O(1) que saltamos.

Pasos	**Complejidad**
Buscar el sector. (En model.)	O(x), x sectores.
Buscar subsector. (En model.)	O(y*log(y)), y subsector (Un merge sort).
Ordenar subsectores (En model.)	O(k*log(k)), k actividades económicas (Un merge sort).
TOTAL	***O(n*log(n))***

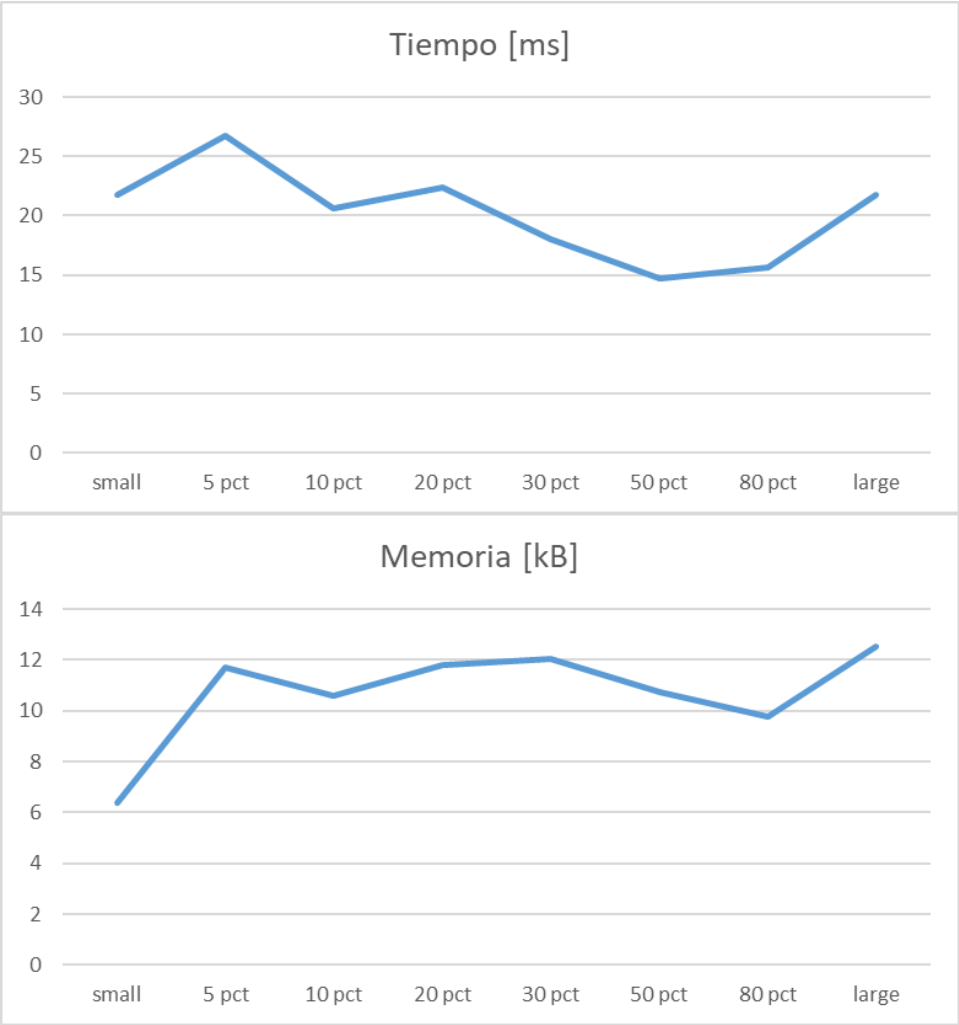
Pruebas Realizadas

Procesadores	**Intel(R) Core(TM) i5-9300H @2.40Ghz**
Memoria RAM	8 GB
Sistema Operativo	Windows 11 Home

Tablas de datos

Req 6	Tiempo (ms)	Memoria (kB)
small	21.754	6.39
5pct	26.787	11.689
10pct	20.593	10.569
20pct	22.404	11.815
30pct	18.061	12.018
50pct	14.754	10.743
80pct	15.623	9.782
large	21.78	12.543

Graficas



Análisis

El algoritmo tiene una complejidad temporal de O(n*log(n)). Buscar el sector relevante es muy fácil, ya que solo tiene que recorrer todos los sectores y elegir el que tiene el mayor total de ingresos netos. Los subsectores son más complicados (en la manera que lo hicimos) porque ordenamos los subsectores antes de encontrar los subsectores adecuados. El ordenamiento es la parte más compleja de este requerimiento, y se hace tres veces. Una cuando estamos ordenando los subsectores, y dos veces ordenando las actividades económicas por los dos subsectores. Cuál toma más tiempo depende de las cantidades de subsectores y actividades económicas.

Requerimiento 7

Listar el TOP (N) de las actividades económicas con el menor total de costos y gastos para un subsector en un año específicos.

Descripción

Entrada	Año, subsector económico, numero top (N)
Salidas	TOP (N) actividades económicas con el menor total de costos y gastos.
Implementado (Sí/No)	Sí/ El grupo.

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo. Hay varios pasos que son O(1) que saltamos.

Pasos	**Complejidad**
Ordenar subsector. (En model.)	O(k*log(k)), k actividades económicas (Un merge sort).
Crear tabla. (En view.)	O(K), K constante.
TOTAL	***O(k*log(k))***

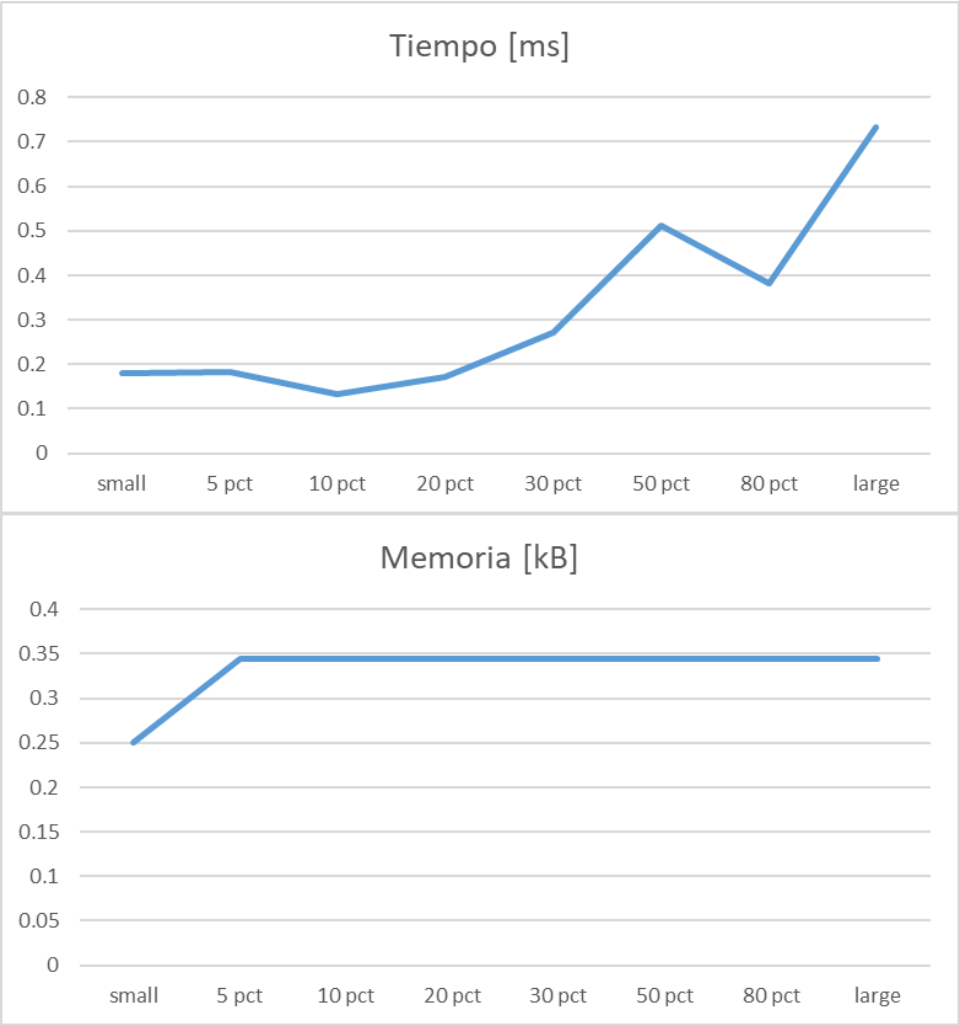
Pruebas Realizadas

Procesadores	**Intel(R) Core(TM) i5-9300H @2.40Ghz**
Memoria RAM	8 GB
Sistema Operativo	Windows 11 Home

Tablas de datos

Req 7	Tiempo (ms)	Memoria (kB)
small	0.179	0.25
5pct	0.182	0.344
10pct	0.132	0.344
20pct	0.172	0.344
30pct	0.271	0.344
50pct	0.513	0.344
80pct	0.383	0.344
large	0.733	0.344

Graficas



Análisis

El algoritmo tiene una complejidad temporal de O(k*log(k)). Con los atributos relevantes ya dentro de los objetos, el algoritmo solo tiene que coger el subsector y ordenarlo. Luego se hace una tabla con los elementos ordenados. El ordenamiento tiene la complejidad más grande, entonces este es la complejidad del algoritmo.

Requerimiento 8

Listar el TOP (N) de actividades económicas de cada subsector con los mayores totales de impuestos a cargo para un año específico

Descripción

Entrada	Año, subsector económico, numero top (N)
Salidas	TOP (N) actividades económicas de cada subsector con los mayores totales de impuestos.
Implementado (Sí/No)	Sí/ El grupo.

Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo. Hay varios pasos que son O(1) que saltamos.

Pasos	**Complejidad**
Ordenar lista de subsectores. (En model.)	O(y*log(y)), y subsectores en total(Un merge sort).
Ordenar cada subsector. (En model.)	O(y*k*log(k)), y subsectores en totay, k actividades económicas.
Crear tablas. (En view.)	O(y), y subsectores en total.
TOTAL	***O(y*k*log(k))***

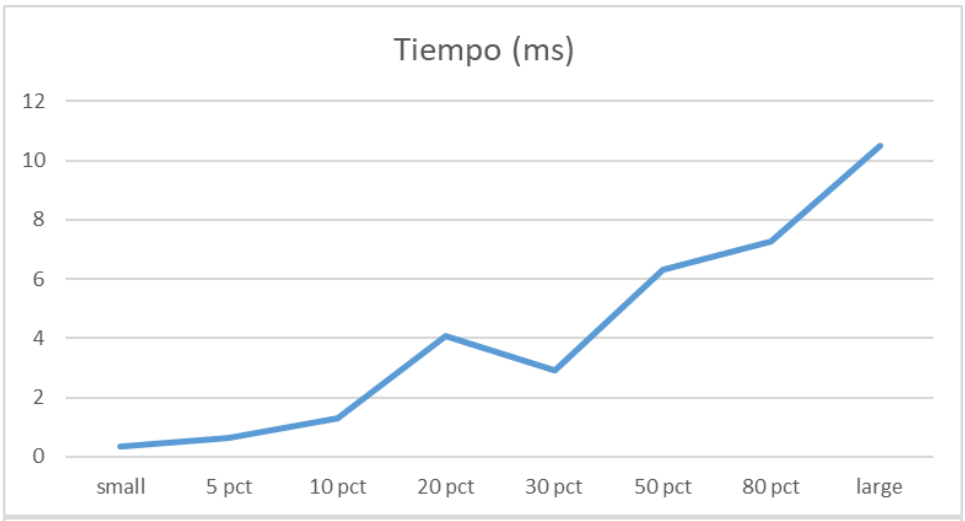
Pruebas Realizadas

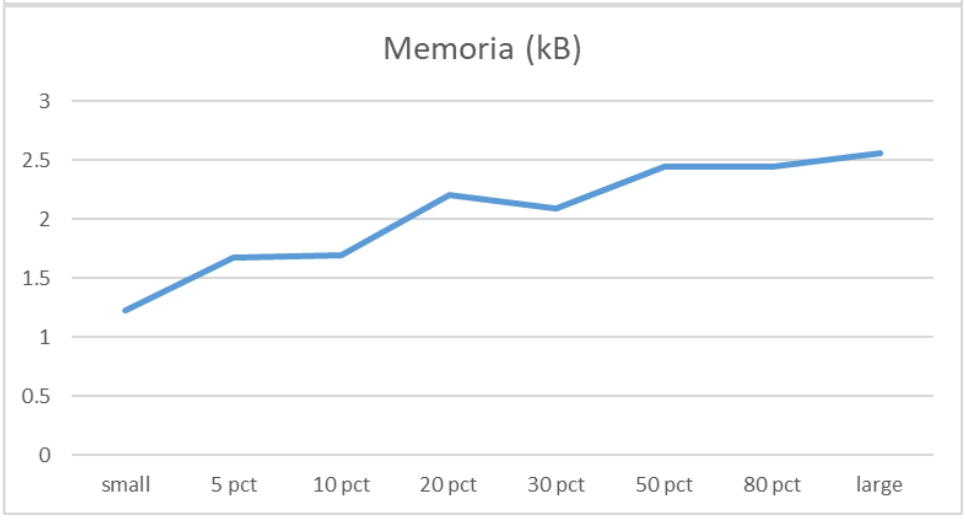
Procesadores	**Intel(R) Core(TM) i5-9300H @2.40Ghz**
Memoria RAM	8 GB
Sistema Operativo	Windows 11 Home

Tablas de datos

Req 8	Tiempo (ms)	Memoria (kB)
small	0.327	1.227
5pct	0.625	1.672
10pct	1.307	1.696
20pct	4.083	2.203
30pct	2.934	2.086
50pct	6.3	2.438
80pct	7.253	2.438
large	10.519	2.555

Graficas





Análisis

El algoritmo tiene una complejidad temporal de O(y*k*log(k)), con y el número de subsectores en total y k la cantidad más grande de actividades económicas dentro de un subsector. El algoritmo hace principalmente 3 cosas. Ordena una lista de los subsectores, ordena cada subsector dentro de esa lista y crea una tabla. La complejidad viene de ordenar un subsector (complejidad O(k*log(k))) y veces. Este es el caso porque cada subsector es ordenado usando merge sort.