**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.

# **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))*** |

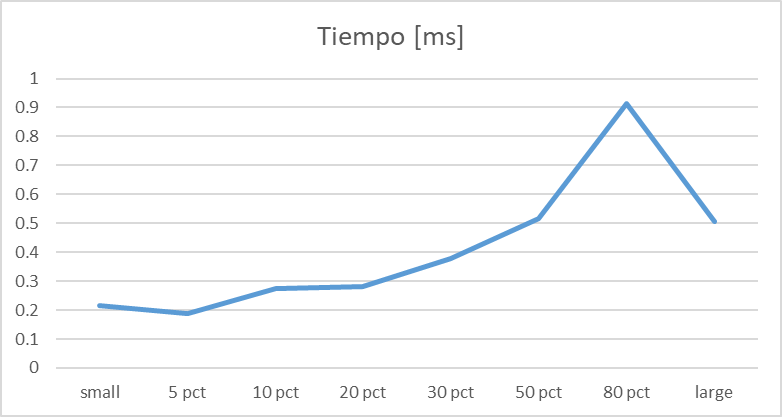
## **Pruebas Realizadas**

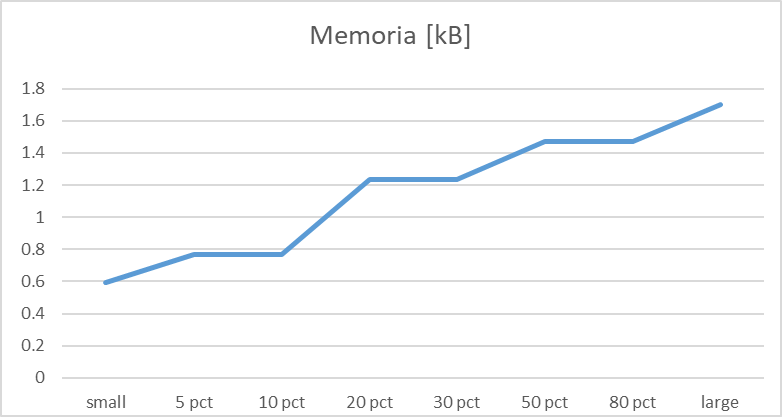
|  |  |
| --- | --- |
| Procesadores | Intel(R) Core(TM) i5-9300H @2.40Ghz |
| Memoria RAM | 8 GB |
| Sistema Operativo | Windows 11 Home |

### **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))*** |

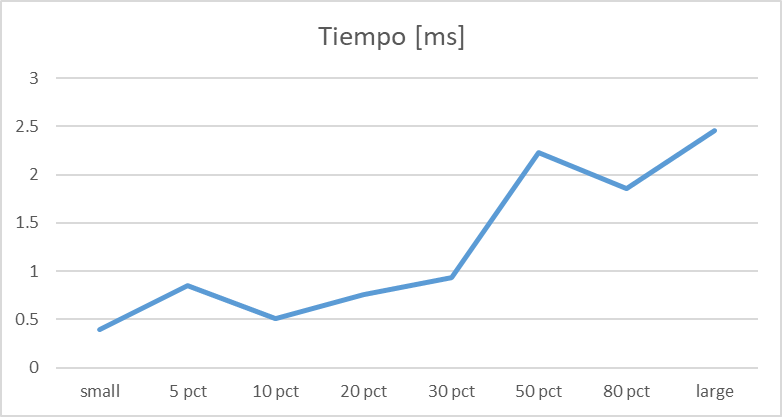
## **Pruebas Realizadas**

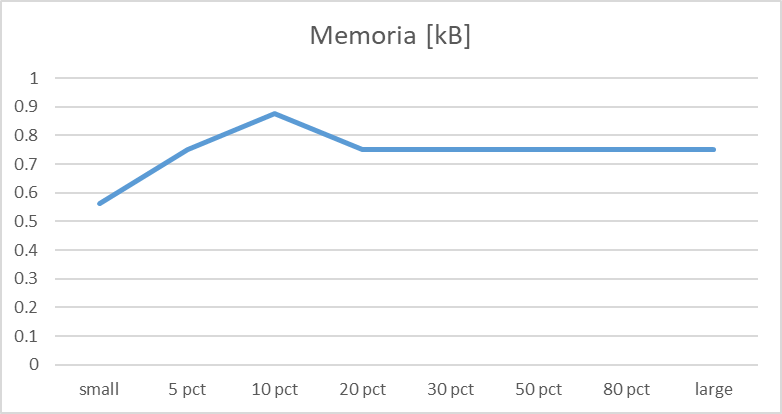
|  |  |
| --- | --- |
| Procesadores | Intel(R) Core(TM) i5-9300H @2.40Ghz |
| Memoria RAM | 8 GB |
| Sistema Operativo | Windows 11 Home |

### **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))*** |

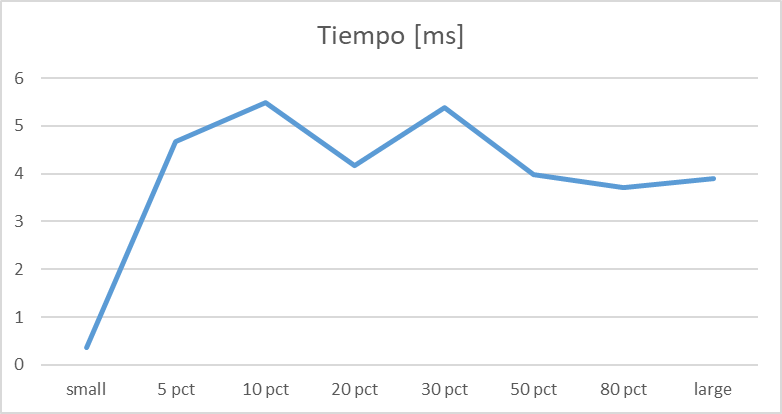
## **Pruebas Realizadas**

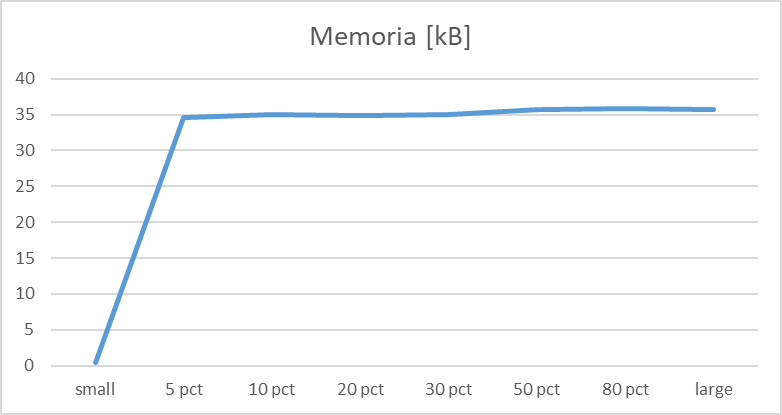
|  |  |
| --- | --- |
| Procesadores | Intel(R) Core(TM) i5-9300H @2.40Ghz |
| Memoria RAM | 8 GB |
| Sistema Operativo | Windows 11 Home |

### **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))*** |

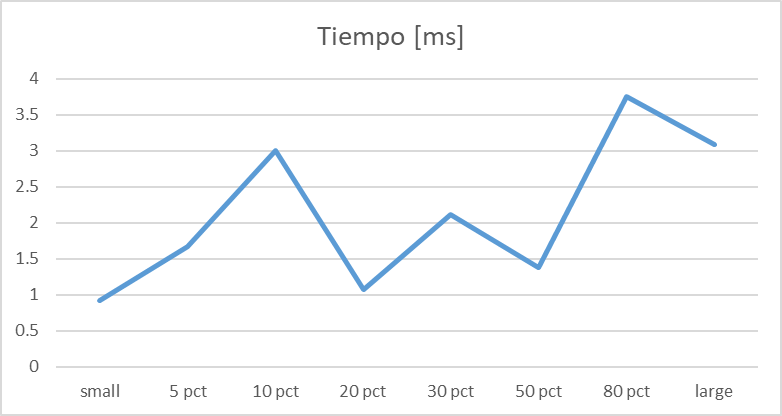
## **Pruebas Realizadas**

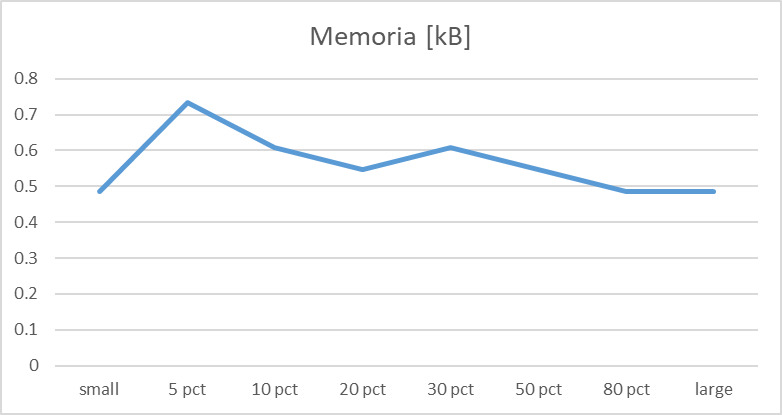
|  |  |
| --- | --- |
| Procesadores | Intel(R) Core(TM) i5-9300H @2.40Ghz |
| Memoria RAM | 8 GB |
| Sistema Operativo | Windows 11 Home |

### **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))*** |

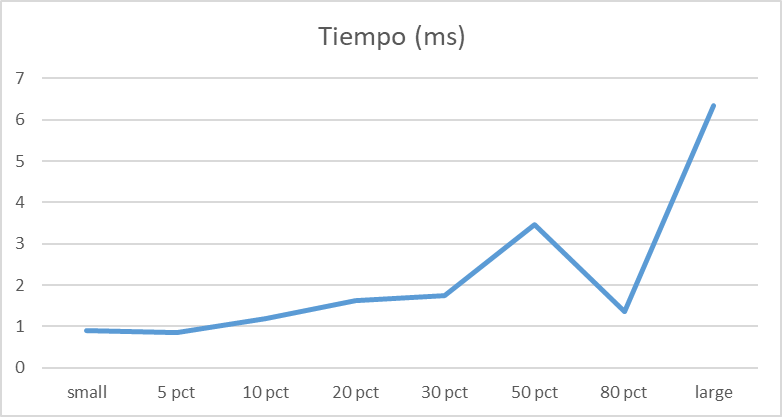
## **Pruebas Realizadas**

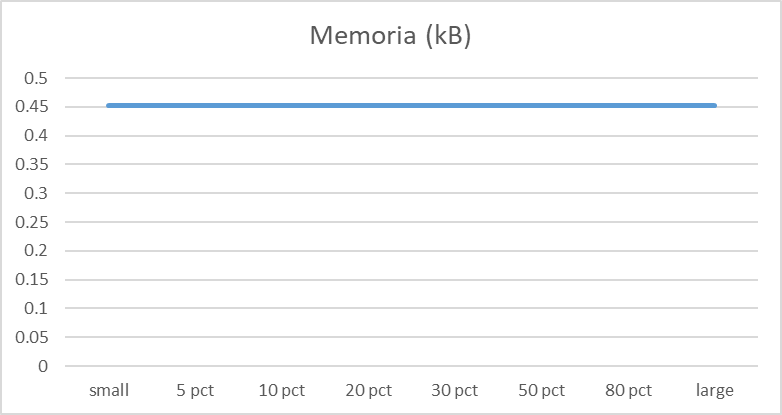
|  |  |
| --- | --- |
| Procesadores | Intel(R) Core(TM) i5-9300H @2.40Ghz |
| Memoria RAM | 8 GB |
| Sistema Operativo | Windows 11 Home |

### **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))*** |

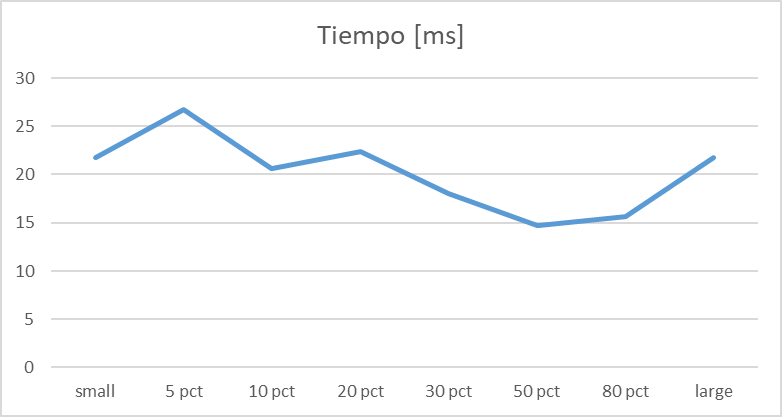
## **Pruebas Realizadas**

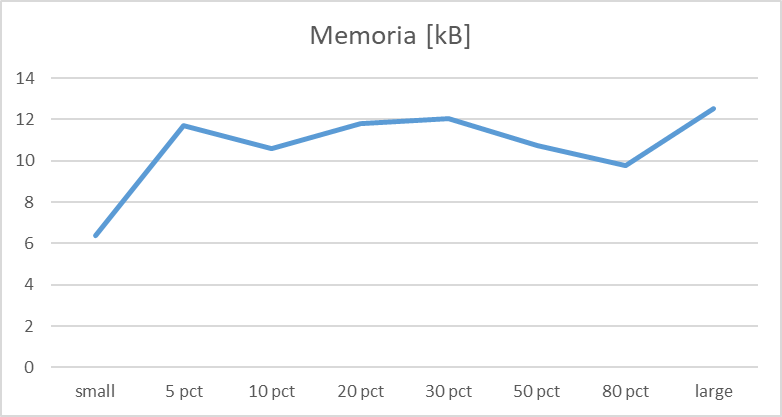
|  |  |
| --- | --- |
| Procesadores | Intel(R) Core(TM) i5-9300H @2.40Ghz |
| Memoria RAM | 8 GB |
| Sistema Operativo | Windows 11 Home |

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

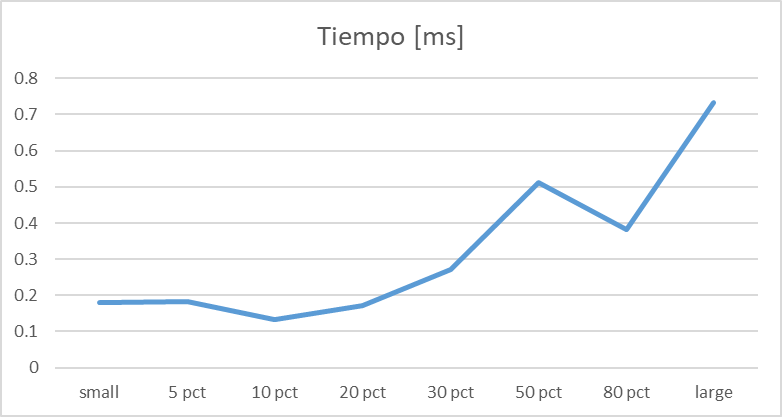
## **Pruebas Realizadas**

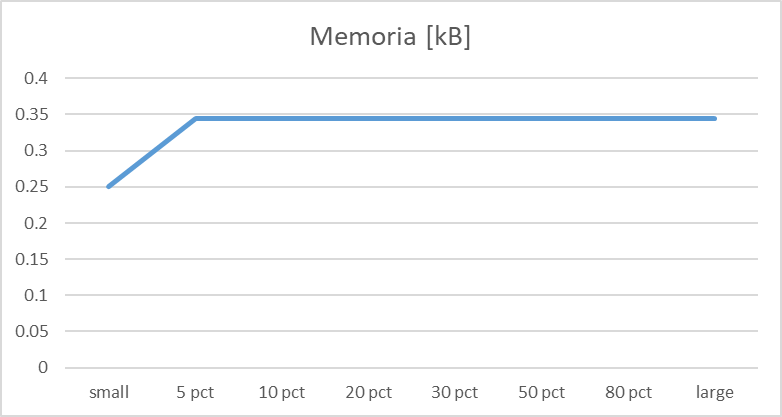
|  |  |
| --- | --- |
| Procesadores | Intel(R) Core(TM) i5-9300H @2.40Ghz |
| Memoria RAM | 8 GB |
| Sistema Operativo | Windows 11 Home |

### **Tablas de datos**

|  |  |  |
| --- | --- | --- |
| Req 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)), 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 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** |
|  |  |
| ***TOTAL*** |  |

## **Pruebas Realizadas**

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

### **Tablas de datos**

### **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.