**ANÁLISIS DEL RETO**

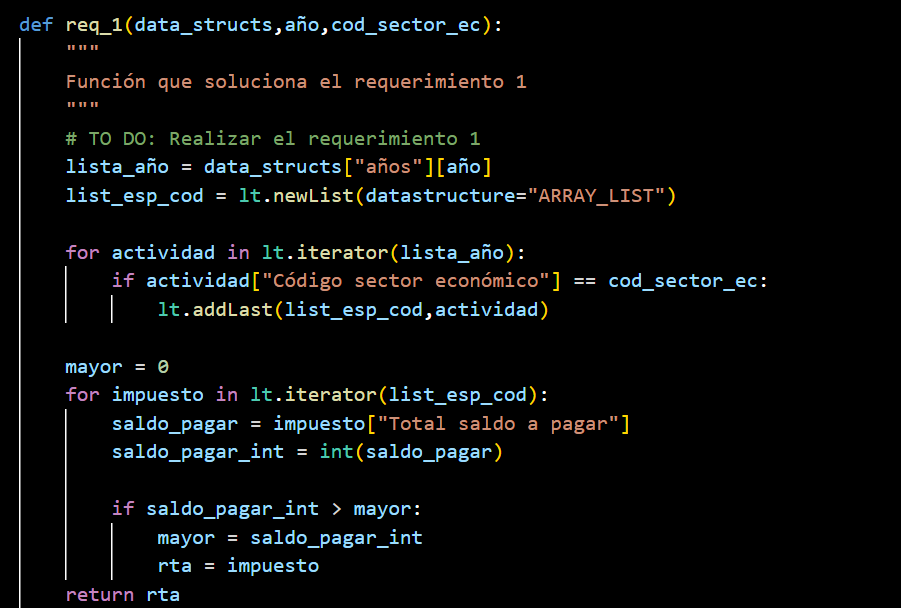
Jacobo Zarruk, 202223913, j.zarruk@uniandes.edu.co

María José Amorocho, 202220179, m.amorocho@uniandes.edu.co

# **Requerimiento 1**

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

## **Descripción**



Para cumplir con el requerimiento primero se obtiene la lista de actividades de ese año. Luego se recorre esta lista para buscar las actividades económicas que coinciden con el sector dado por parámetro, y posteriormente se comparan por su saldo a pagar para obtener la actividad con mayor valor.

|  |  |
| --- | --- |
| **Entrada** | Año, código sector económico, estructura de datos |
| **Salidas** | Actividad económica con mayor saldo a pagar perteneciente al año y código económico establecidos |
| **Implementado (Sí/No)** | Implementado por María José |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Acceder a la lista de actividades económicas del año específico | O(1) |
| Crear una lista para guardar las actividades económicas de ese año con el código del sector económico buscado | O(1) |
| Recorrer la lista de actividades económicas del año. Si su código de sector económico es el mismo que el código dado por parámetro, se agrega a la lista creada para almacenar las actividades económicas con el sector económico buscado. | O(n) |
| Se recorre la lista que tiene las actividades con el mismo código del sector económico buscado por parámetro para comprar cada impuesto por su [“Total saldo a pagar”]. Si el saldo a pagar es mayor a la variable “mayor”, se establece esa actividad como respuesta (y se actualiza el valor de “mayor”). | 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).

Las pruebas realizadas fueron realizadas en una maquina con las siguientes especificaciones. El año dado por parámetro fue 2016 usando el sector 1.

|  |  |
| --- | --- |
| Procesadores | Intel(R) Core(TM) i5-10210U CPU @1.60GHz 2.10 GHz |
| Memoria RAM | 8 GB |
| Sistema Operativo | Windows 10 |

|  |  |  |
| --- | --- | --- |
| **Entrada** | **Tiempo (ms)** | **Memoria** |
| small | 0,18 | 1,48 |
| 5 pct | 0,22 | 1,48 |
| 10 pct | 0,19 | 1,48 |
| 20 pct | 0,24 | 1,48 |
| 30 pct | 0,30 | 1,48 |
| 50 pct | 0,37 | 1,48 |
| 80 pct | 0,45 | 1,48 |
| large | 0,79 | 1,48 |

### **Tablas de datos**

Las tablas con la recopilación de datos de las pruebas.

|  |  |  |
| --- | --- | --- |
| **Muestra** | **Salida (código actividad económica)** | **Tiempo (ms)** |
| small | 130 | 0,18 |
| 5 pct | 125 | 0,22 |
| 10 pct | 125 | 0,19 |
| 20 pct | 125 | 0,24 |
| 30 pct | 125 | 0,30 |
| 50 pct | 125 | 0,37 |
| 80 pct | 125 | 0,45 |
| large | 125 | 0,79 |

### **Graficas**

Las gráficas con la representación de las pruebas realizadas.

## **Análisis**

Análisis de resultados de la implementación, tener cuenta las pruebas realizadas y el análisis de complejidad.

# **Requerimiento 3**

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

## **Descripción**



Para cumplir con el requerimiento se busca la lista de actividades económicas correspondiente a un año específico. La información de esta lista se ordena en un mapa que tiene como llaves el numero de un subsector económico, y como valor una lista de actividades económicas de ese subsector. De cada lista se obtiene la suma de retenciones de las actividades económicas y se obtiene suma de menor valor junto con el número del sector económico al que pertenece esta cantidad. Posteriormente se obtienen los diferentes totales del subsector que son solicitados y se toman las actividades de mayor y menor aporte al total de retenciones.

|  |  |
| --- | --- |
| **Entrada** | Estructura de datos, año |
| **Salidas** | Tupla con la información del subsector económico con menores retenciones en el año dado y las actividades económicas con mayores y menores aportes al total de retenciones. |
| **Implementado (Sí/No)** | Si. Implementado por María José |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Obtener la lista de actividades económicas pertenecientes a un año específico. | O(1) |
| Se llama a la función Sort\_list\_year\_subsector que recorre la lista de actividades económicas en un año y las organiza en un mapa (map\_sort\_subsector). Las llaves de estos mapas corresponden al número de un subsector económico y como valor tienen una lista de las actividades económicas con ese mismo subsector. | O(1) |
| Se crea el mapa map\_subsectores\_y\_totales que tendrá como llave el código de un subsector económico y como valor el total de retenciones para ese subsector. | O(1) |
| Se recorre el mapa map\_sort\_subsector. Por cada subsector se toma su valor (es decir, una lista con las actividades económicas de este subsector) y se llama a la función sumar\_elementos, que recorre una lista dada sumando un criterio brindado por parámetro. En este caso se suman el total de retenciones de cada actividad económica por cada subsector en el mapa. Cuando se obtiene el total de retenciones de un subsector, se añade al mapa map\_subsectores\_y\_totales. Como llave, este valor tendrá el número del subsector económico que se recorrió. | O(n) |
| Se llama a la función mayor\_o\_menor\_mapa. Esta función retorna el mayor o el menor valor de un mapa (según lo que se indique) y su llave correspondiente (en una tupla). La función mayor\_o\_menor\_mapa se aplica sobre map\_subsectores\_y\_totales para obtener el código del subsector económico con mayor total de retenciones y el valor numérico de esta cantidad. | O(1) |
| Habiendo obtenido el código del subsector económico con menor retenciones, se busca la lista de actividades asociadas a este en el mapa map\_sort\_subsector, usando las funciones mp.get() y me.getValue(). La lista de actividades económicas se ordena ascendentemente con respecto al total de retenciones de cada una. | O(1) |
| Se llama a la función obtener\_totales\_subsector para calcular los totales solicitados del subsector con menor total de retenciones. obtener\_totales\_subsector se encarga de recorrer las actividades económicas de una lista y sumar el total de ingresos netos, costos y gastos, saldo por pagar y saldo a favor de cada una de estas. Estos Totales son guardados en un diccionario, que tiene como llaves el título del parámetro calculado (eg. “Total ingresos netos”) y como valor la suma obtenida de este criterio. El diccionario con esta información se guarda en la variable info\_totales\_subsect. | O(1) |
| Para obtener las actividades con mayores y menores aportes al total de retenciones se toma la lista de actividades económicas ordenada (lista\_act\_sub\_sec\_menor\_ret) que contiene las actividades económicas pertenecientes al subsector con mayores retenciones. Si el tamaño de la lista es menor a 6, se retorna la lista ordenada. Si tiene más de 6 elementos se llama a la función get\_3\_last\_and\_first\_list que retorna una lista con los tres primeros y 3 últimos elementos de una lista dada por parámetro. | O(1) |
| Se retorna el diccionario con la información de totales del sector económico con menos retenciones y una lista de las actividades económicas con mayores y menores aportes en una tupla. | O(n) |
| ***TOTAL*** | ***O(n)*** |

## **Pruebas Realizadas**

Las pruebas realizadas fueron realizadas en una maquina con las siguientes especificaciones. Como año se tomó el 2013

|  |  |
| --- | --- |
| Procesadores | Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz 2.10 GHz |
| Memoria RAM | 8 GB |
| Sistema Operativo | Windows 10 |

|  |  |  |
| --- | --- | --- |
| **Entrada** | **Tiempo (ms)** | **Memoria** |
| small | 6,80 | 145,47 |
| 5 pct | 13,06 | 147,52 |
| 10 pct | 11,46 | 147,54 |
| 20 pct | 15,83 | 150,52 |
| 30 pct | 19,54 | 151,22 |
| 50 pct | 30,82 | 152,14 |
| 80 pct | 54,80 | 152,67 |
| large | 45,13 | 155,51 |

### **Tablas de datos**

Las tablas con la recopilación de datos de las pruebas.

|  |  |  |
| --- | --- | --- |
| **Muestra** | **Salida (total retenciones subsector)** | **Tiempo (ms)** |
| small | 1190 | 6,80 |
| 5 pct | 13 | 13,06 |
| 10 pct | 13 | 11,46 |
| 20 pct | 13 | 15,83 |
| 30 pct | 13 | 19,54 |
| 50 pct | 3734 | 30,82 |
| 80 pct | 17 | 54,80 |
| large | 7 | 45,13 |

### **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.