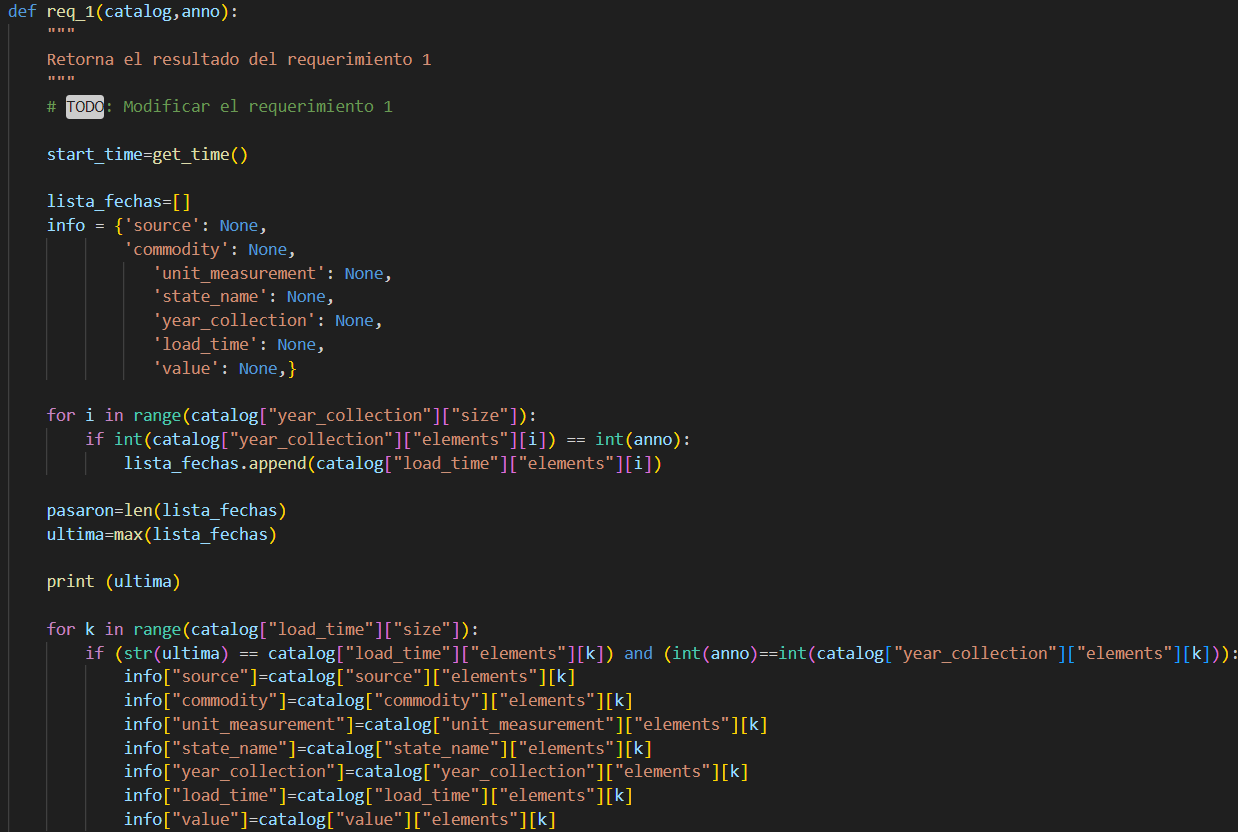
**ANÁLISIS DEL RETO**

# Samuel Villamil Alvarez, s.villamila@uniandes.edu.co, 202421118

# Juan Diego García, jd.garcia12@uniandes.edu.co, 202423575

# Tomás Aponte, t.aponte@uniandes.edu.co, 202420148

# **Requerimiento <<1>>**



## **Descripción**

Este requerimiento se encarga de buscar y mostrar al usuario el ultimo registro encontrado durante un año especifico. Además, retorna sus características base. Lo que hace es crear un listado de fechas en donde se insertara la información de los registros que cumplan el criterio. Para encontrar dichos registros se ejecuta un ciclo el cual compara la información necesaria con la de todos los registros para así añadir los que cumplen el criterio. Con el primer ciclo encuentra la longitud de la lista de los registros que cumplen los requisitos y la fecha con el valor más alto de estos. El segundo busca el registro que cumple con esta fecha de subida y el año adecuado.

|  |  |
| --- | --- |
| **Entrada** | Catálogo, año. |
| **Salidas** | Tiempo de ejecución, numero de registros que se registraron ese año, año de recopilación, fecha de carga, origen del registro, departamento, producto, unidad y valor. |
| **Implementado (Sí/No)** | Si, implementado por Samuel Villamil. |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Creación de lista para fechas y diccionario con la estructura para los datos. | O(1) |
| Se recorre el catálogo en búsqueda del tiempo de carga para los datos en el año definido por el usuario. | O(n) |
| Se definen variables para el tamaño de la muestra de datos y el dato con el valor más alto para la lista de fechas. | O(1) |
| Se recorre el catalogo a medida del tamaño de la muestra de la fecha de subida, para encontrar la información de el ultimo registro a partir de la variable “ultimo” y el año introducido por el usuario. | O(n) |
| ***TOTAL*** | ***O(n)*** |

## **Pruebas Realizadas**

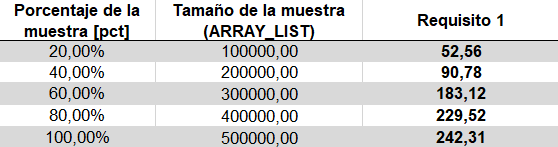
Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

La fecha utilizada fue: 2012

|  |  |
| --- | --- |
| **Entrada** | **Tiempo (s)** |
| 20% | 52.56ms |
| 40% | 90.78ms |
| 60% | 183.12ms |
| 80% | 229.52ms |
| 100% | 242.31ms |

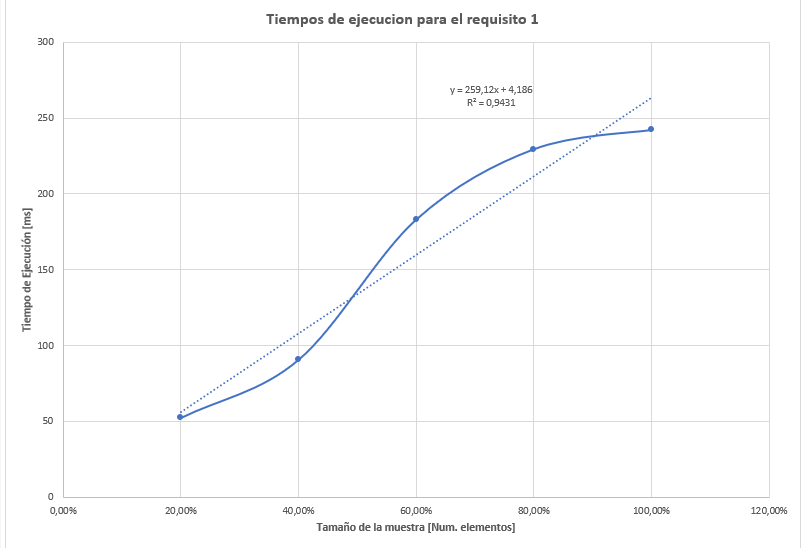
### **Tablas de datos**

Las tablas con la recopilación de datos de las pruebas.



### **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.

En la gráfica, se puede evidenciar una correlación lineal en cuanto a el tiempo de ejecución de la función y el tamaño de la muestra, lo cual, nos indica una implementación lineal o O(n) que coincide con lo esperado según el código.

# **Requerimiento <<2>>**

## **Descripción**

Este requerimiento opera de la misma manera del primero. Crea una lista para los departamentos que coincidan con el criterio, así como un diccionario para almacenar la información. Después adjunta todos los registros con el estado correspondiente a la lista y obtiene las variables de la longitud de la lista y el valor máximo de la misma. Después hace otro recorrido para encontrar el registros con el debido tiempo de subida y departamento, para finalmente devolver la información obtenida.

|  |  |
| --- | --- |
| **Entrada** | Catálogo, departamento. |
| **Salidas** | Tiempo de ejecución, numero de registros que se registraron para el departamento, año de recopilación, fecha de carga, origen del registro, departamento, producto, unidad y valor. |
| **Implementado (Sí/No)** | Si, implementado por Samuel Villamil. |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Creación de lista para fechas y diccionario con la estructura para los datos. | O(1) |
| Se recorren todos los departamentos en el catálogo en búsqueda de fechas de carga del departamento que especifico el usuario. | O(n) |
| Se definen variables para el tamaño de la muestra de datos y el dato con el valor más alto para la lista de fechas. | O(1) |
| Se recorre el catalogo en busca de el registro que coincida con la información de tiempo de carga máxima y nombre del estado que deseado para extraer su información al diccionario info. | 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).

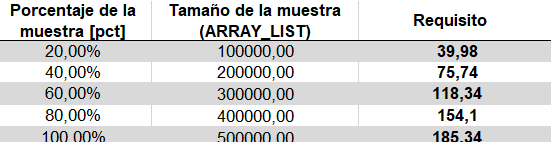
El parámetro utilizado fue: CALIFORNIA

|  |  |
| --- | --- |
| Procesador | AMD Ryzen 5 3600 |
| Memoria RAM | 16 GB |
| Sistema Operativo | Windows 11 |

|  |  |
| --- | --- |
| **Entrada** | **Tiempo (s)** |
| 20% | 39.98ms |
| 40% | 75.74ms |
| 60% | 118.24ms |
| 80% | 154.10ms |
| 100% | 185.34ms |

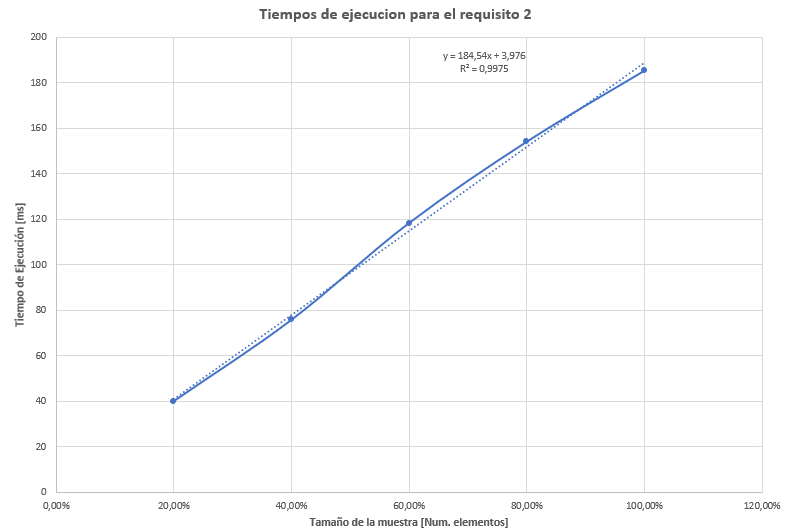
### **Tablas de datos**

Las tablas con la recopilación de datos de las pruebas.



### **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.

En esta grafica es posible evidenciar una relación O(n) en cuanto a el tiempo de ejecución y el tamaño de la muestra. Esto es esperado gracias a la estructura del código implementado y la similitud de esta función a la anterior.

# **Requerimiento <<3>>**



## **Descripción**

Este requerimiento se encarga de recopilar todos los registros con su respectiva información en una lista, dentro de un margen de tiempo para un departamento especificado por el usuario, indicando también cuantos de los orígenes de los registros son de tipo “*survey*” o *“census*”. Si esta lista supera los 20 elementos, se retorna solo los primeros y últimos 5. Además, si el margen de tiempo no es coherente, se retorna None.

|  |  |
| --- | --- |
| **Entrada** | Catalogo con los datos del dataframe, el departamento deseado y el intervalo de tiempo deseado (año inicial / año final) |
| **Salidas** | Lista con los datos de los registros encontrados, total de datos de tipo “*survey*”, total de datos de tipo *“census*”, total de datos que pasaron el filtro de búsqueda y el tiempo de ejecución del requisito en milisegundos |
| **Implementado (Sí/No)** | Si, implementado por Samuel Villamil |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Validación del intervalo (if anno\_f<anno\_i) | O(1) |
| Recorrido total del catalogo (for i in range …) | O(n) |
| Filtro de búsqueda (if … and …) | O(n) |
| Asignación info | O(1) |
| Añadir a lista | O(n) |
| Verificación del tamaño pasaron | O(1) |
| Posible agregado a nova\_lista | O(n) |
| ***TOTAL*** | ***O(n)*** |

## **Pruebas Realizadas**

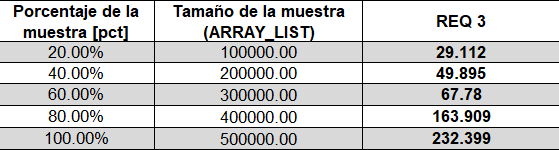
Las pruebas realizadas fueron realizadas en una maquina con las siguientes especificaciones:

|  |  |
| --- | --- |
| Procesadores | Intel® Core™ i7-10610U CPU @ 1.80Hz |
| Memoria RAM | 16 GB |
| Sistema Operativo | Windows 11 |

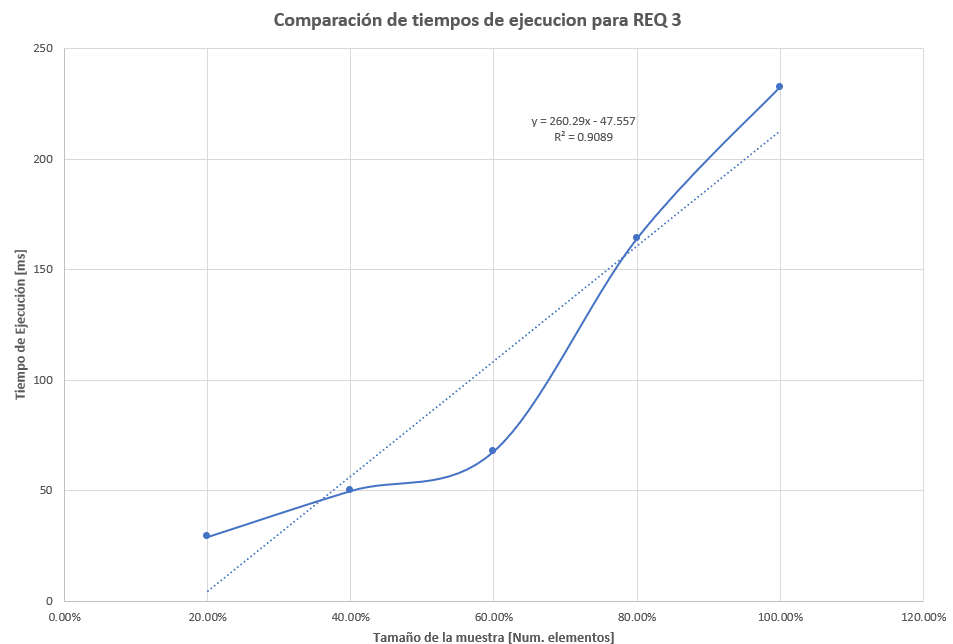
Los datos de entrada fueron: ILLINOIS, 2001, 2002.

|  |  |
| --- | --- |
| **Entrada** | **Tiempo (s)** |
| 20% | **29.112** |
| 40% | **49.895** |
| 60% | **67.78** |
| 80% | **163.909** |
| 100% | **232.399** |

### **Tablas de datos**



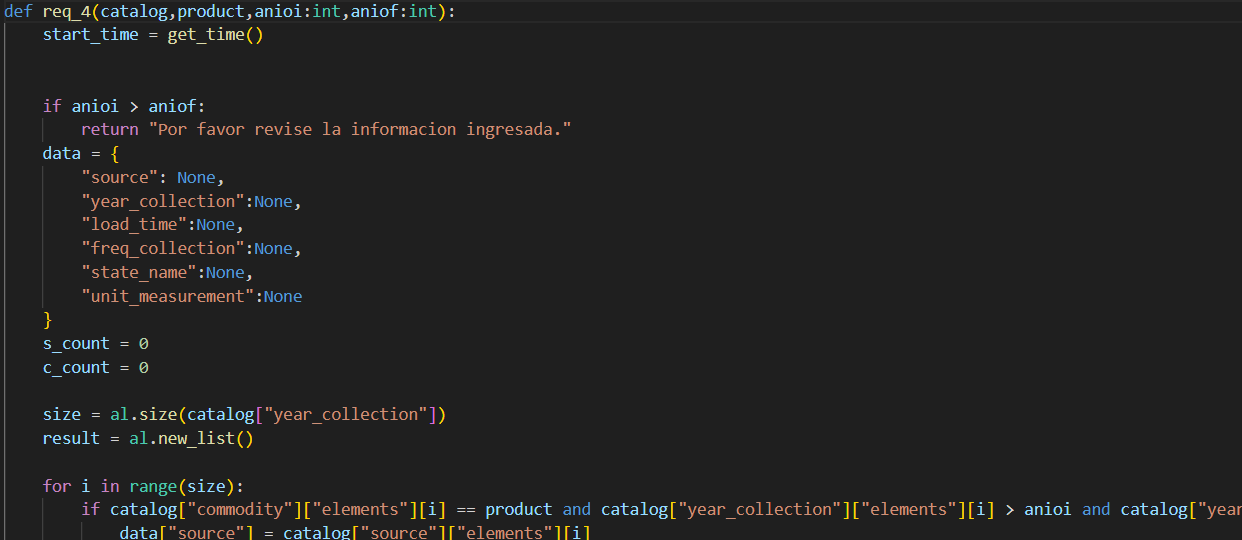
### **Graficas**



## **Análisis**

La complejidad para este requisito, teniendo en cuenta el uso de array\_list dentro del mismo, tiene sentido que sea de tipo lineal O(n), pues a medida que aumentan los datos, el tiempo de ejecución de la función aumenta de manera proporcional a estos, debido a que la necesidad, en el peor de los casos, de recorrer la totalidad de la lista. Lo anterior además se puede evidenciar de manera experimental en la gráfica, pues la ecuación ajustada de la misma sigue este comportamiento lineal, y además la “aceleración” que se presenta con el 80% de los datos también representa ese peor caso mencionado anteriormente.

# **Requerimiento <<4>>**



## **Descripción**

Este requerimiento recorre el catálogo buscando la posición de los registros que contienen el producto deseado y pertenecen a un rango de tiempo especificado para recolectar su información en una lista. El requerimiento también cuenta la cantidad de registros de tipo SURVEY y CENSUS. En caso de que el tamaño de la lista sea mayor a 20 elementos, esta solamente mostrara los primeros y últimos 5.

|  |  |
| --- | --- |
| **Entrada** | Catálogo, producto, año inicial, año final. |
| **Salidas** | Registros encontrados, total de survey y census, total de datos que pasaron el filtro de búsqueda, tiempo de ejecución. |
| **Implementado (Sí/No)** | Si, implementado por Tomas Aponte |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Se comprueba la validez del intervalo | O(1) |
| Se recorre el catalogo en busca de registros pertinentes | O(n) |
| Contador de survey o census | O(n) |
| Verificación del tamaño count | O(1) |
| Posible cambio de result a short\_result | O(n) |
| ***TOTAL*** | ***O(n)*** |

## **Pruebas Realizadas**

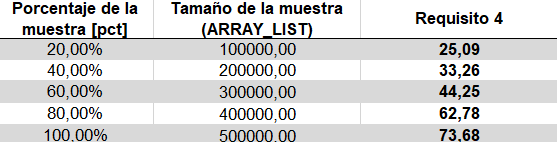
Las pruebas realizadas fueron realizadas en una maquina con las siguientes especificaciones:

|  |  |
| --- | --- |
| Procesadores | AMD Ryzen 5 3600 |
| Memoria RAM | 16 GB |
| Sistema Operativo | Windows 11 |

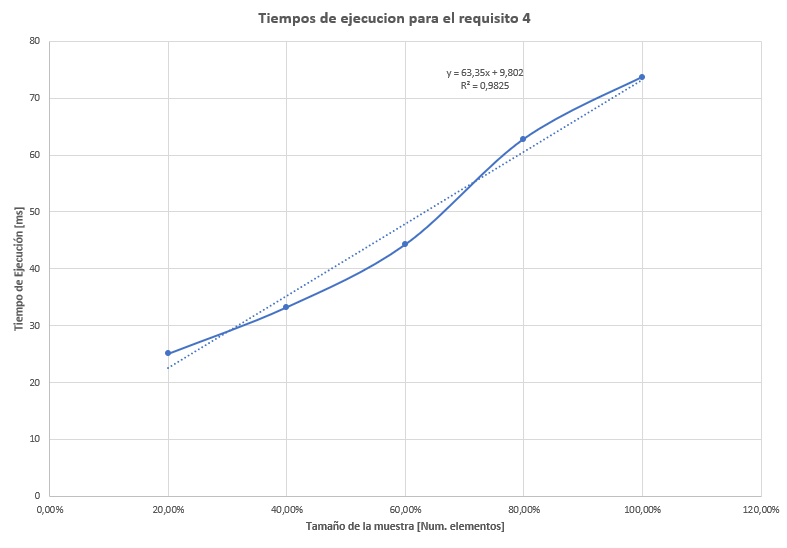
Los datos de entrada fueron: CATTLE, 2001, 2002.

|  |  |
| --- | --- |
| **Entrada** | **Tiempo (ms)** |
| 20% | **25.09ms** |
| 40% | **33.26ms** |
| 60% | **44.25ms** |
| 80% | **62.78ms** |
| 100% | **73.68ms** |

### **Tablas de datos**



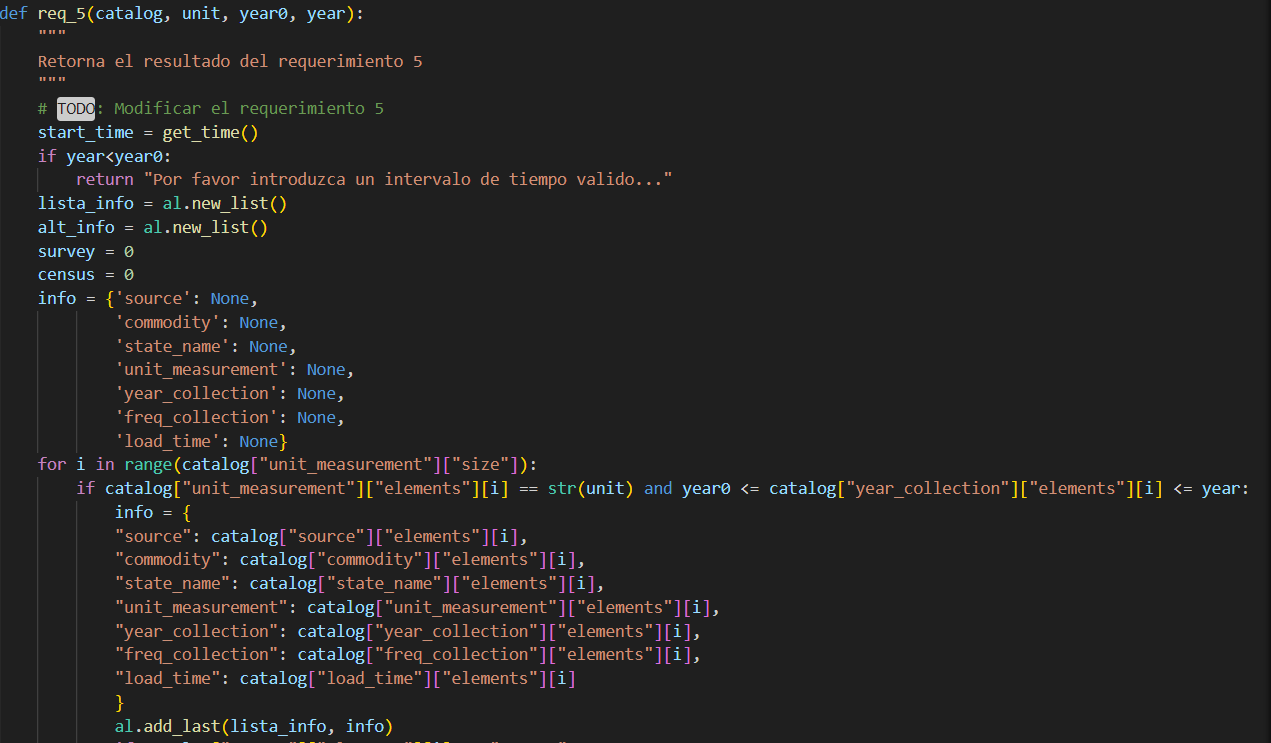
### **Graficas**



## **Análisis**

La similitud entre la implementación de esta función y la anterior es clara, por lo que se esperaría que la complejidad temporal se comportara de la misma manera, lo cual es correcto en este caso. En la gráfica se nota claramente la relación lineal entre el tiempo de ejecución de la función y el tamaño de la muestra de datos por lo que se puede concluir una complejidad O(n).

# **Requerimiento <<5>>**



## **Descripción**

Este requerimiento se encarga de recopilar todos los registros con su respectiva información en una lista, dentro de un margen de tiempo para un tipo de unidad de valor asociado especificado por el usuario, indicando también cuantos de los orígenes de los registros son de tipo “*survey*” o *“census*”. Si esta lista supera los 20 elementos, se retorna solo los primeros y últimos 5. Además, si el margen de tiempo no es coherente, se retorna None.

|  |  |
| --- | --- |
| **Entrada** | Catalogo con los datos del dataframe, La unidad de valor asociado deseado y el intervalo de tiempo deseado (año inicial / año final) |
| **Salidas** | Lista con los datos de los registros encontrados, total de datos de tipo “*survey*”, total de datos de tipo *“census*”, total de datos que pasaron el filtro de búsqueda y el tiempo de ejecución del requisito en milisegundos |
| **Implementado (Sí/No)** | Si, implementado por Juan García |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Validación del intervalo (if year<year0) | O(1) |
| Recorrido total del catalogo (for i in range …) | O(n) |
| Filtro de búsqueda (if … and …) | O(n) |
| Asignación info | O(1) |
| Añadir a lista\_info | O(n) |
| Verificación del tamaño pasaron | O(1) |
| Posible agregado a alt\_list | O(n) |
| ***TOTAL*** | ***O(n)*** |

## **Pruebas Realizadas**

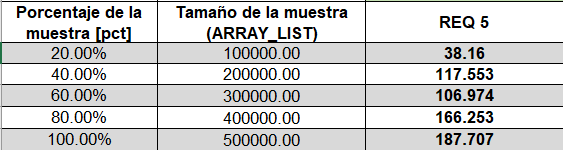
Las pruebas realizadas fueron realizadas en una maquina con las siguientes especificaciones:

|  |  |
| --- | --- |
| Procesadores | AMD Ryzen 7 PRO 2700U w/ Radeon Vega Mobile Gfx |
| Memoria RAM | 16 GB |
| Sistema Operativo | Windows 10 |

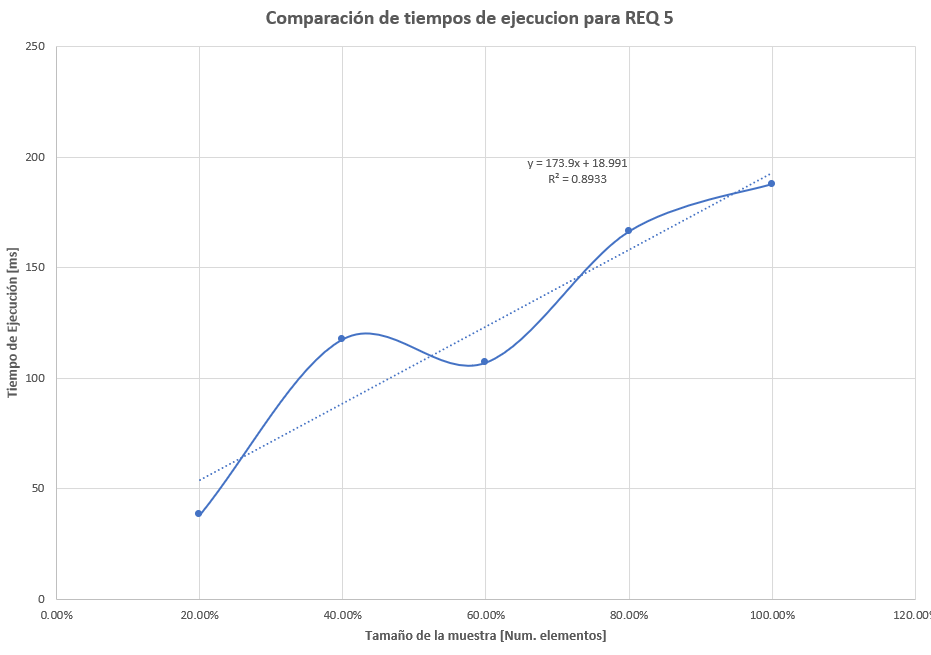
Los datos de entrada fueron: OPERATIONS, 2001, 2002.

|  |  |
| --- | --- |
| **Entrada** | **Tiempo (s)** |
| 20% | **38.16** |
| 40% | **117.553** |
| 60% | **106.974** |
| 80% | **166.253** |
| 100% | **187.707** |

### **Tablas de datos**



### **Graficas**



## **Análisis**

De un mismo modo que con el requisito 4, debido a las similitudes en cuanto a código presentes no es sorpresa que este requisito también tenga un comportamiento lineal O(n). Al analizar la gráfica proporcionada, a pesar de que el ajuste nos permite evidenciar de una manera muy clara este comportamiento, la gráfica fluctúa bastante, reduciendo su tiempo de respuesta con el 60% de los datos para luego aumentar considerablemente el mismo con el 80%. Debido a lo anterior, se puede llegar a la misma conclusión sobre la ausencia o no del peor caso cuando se recorre la lista.

# **Requerimiento <<6>>**

## **Descripción**

Breve descripción de como abordaron la implementación del requerimiento

|  |  |
| --- | --- |
| **Entrada** | Catalogo, fecha inicial, fecha final, departamento. |
| **Salidas** | Tiempo de ejecución, numero de registros que cumplen el criterio, numero de registros de origen CENSUS y SURVEY, tipo de fuente, año de recopilación, fecha de carga, frecuencia de recopilación, departamento, unidad, producto. |
| **Implementado (Sí/No)** | Si, implementado por Samuel Villamil |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Se comprueba la validez de las fechas | O(1) |
| Se crean contadores para ambos SURVEY y CENSUS, un diccionario con la estructura para la información y una lista. | O(1) |
| Se recorre el diccionario en búsqueda de registros que contengan el estado especificado, así como una fecha dentro del rango establecido y se agrega cada uno de los registros a la lista creada. | O(n) |
| En el caso de obtener mas de 20 registros se ejecuta un argumento que almacena los primeros y últimos 5 elementos de la lista por practicidad. | O(1) |
| ***TOTAL*** | ***O(n)*** |

## **Pruebas Realizadas**

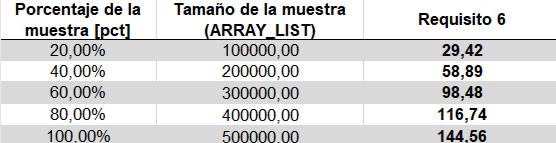
Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

Se uti

|  |  |
| --- | --- |
| **Entrada** | **Tiempo (s)** |
| 20% | 29.42ms |
| 40% | 58.89ms |
| 60% | 98.48ms |
| 80% | 116.74ms |
| 100% | 144.56ms |

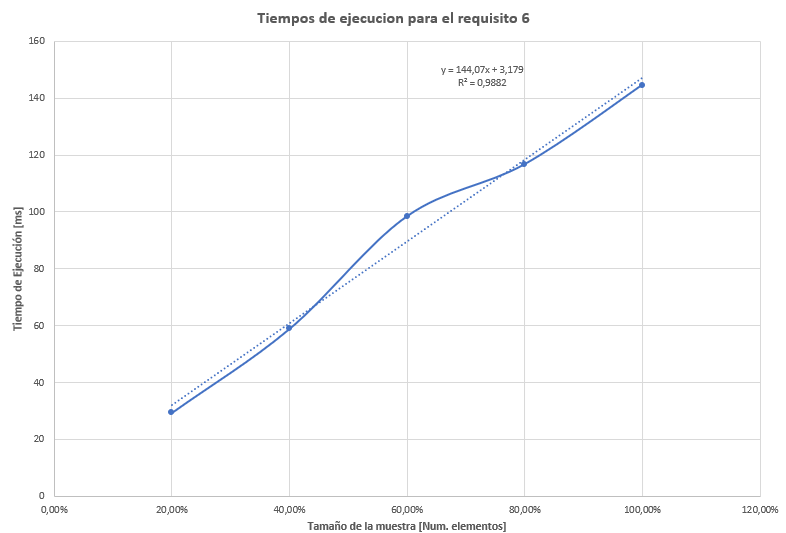
### **Tablas de datos**

Las tablas con la recopilación de datos de las pruebas.



### **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.

Como fue esperado, la gráfica de la función correspondiente al requisito 6, también muestra una correlación lineal, lo que implica una complejidad temporal de O(n). Esto gracias a que, en el peor caso, recorre la lista completamente en búsqueda de un registro especifico.

# **Requerimiento <<7>>**

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

## **Descripción**

Breve descripción de como abordaron la implementación del requerimiento

|  |  |
| --- | --- |
| **Entrada** | Parámetros necesarios para resolver el requerimiento. |
| **Salidas** | Respuesta esperada del algoritmo. |
| **Implementado (Sí/No)** | Si se implementó y quien lo hizo. |

## **Análisis de complejidad**

Análisis de complejidad de cada uno de los pasos del algoritmo

|  |  |
| --- | --- |
| **Pasos** | **Complejidad** |
| Paso 1 | O(...) |
| Paso 2 | O(...) |
| Paso …. | O(...) |
| ***TOTAL*** | ***O(...)*** |

## **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 (s)** |
|  |  |
|  |  |
|  |  |

### **Tablas de datos**

Las tablas con la recopilación de datos de las pruebas.

### **Graficas**

Las gráficas con la representación de las pruebas realizadas.

## **Análisis**

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

# **Requerimiento Ejemplo**

## **Descripción**



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 |

### **Tablas de datos**

Las tablas con la recopilación de datos de las pruebas.

|  |  |  |
| --- | --- | --- |
| **Muestra** | **Salida** | **Tiempo (ms)** |
| small | Dato1 | 0.05 |
| 5 pct | Dato2 | 0.33 |
| 10 pct | Dato3 | 1.28 |
| 20 pct | Dato4 | 2.54 |
| 30 pct | Dato5 | 4.98 |
| 50 pct | Dato6 | 7.51 |
| 80 pct | Dato7 | 13.81 |
| large | Dato8 | 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.