ANÁLISIS DEL RETO 2

Verónica Isaza, 202220917, v.isaza@uniandes.edu.co

Tomás Iriarte, 202220915, t.iriarte@uniandes.edu.co

Juan Esteban Arboleda, 202011354, j.arboledam@uniandes.edu.co

Tabla de contenido

Descripción general del análisis 3
Sección 1: Carga de datos – Análisis teórico 3
Estructura utilizada 3
Análisis de complejidad espacial de la carga 5
Sección 2: Requerimientos – Análisis teórico 7
Requerimientos básicos (1 y 2) 7
Requerimientos individuales (3, 4 y 5) 8
Requerimiento 6 10
Requerimiento 7 12
Requerimiento 8 (bono) 14
Sección 3: Pruebas de ejecución – Tablas y graficas 16
Máquinas utilizadas 16
Pruebas de la carga de datos 16
Pruebas requerimientos básicos (1 y 2) 21
Pruebas requerimientos individuales (3, 4 y 5) 23
Pruebas requerimientos avanzados (6, 7 y 8) 25
Conclusión 28

Descripción general del análisis

El análisis a continuación se divide en tres secciones principales. En primer lugar, se hace una descripción detallada de cómo se hizo la carga de datos, justificando la elección de la estructura de datos utilizada y haciendo un análisis de tiempo y espacio de la carga. A continuación, se hace un análisis teórico de qué es lo que se hace en cada uno de los requerimientos pedidos, que incluye una descripción de estos y su respectivo análisis de complejidad temporal. En la tercera sección se muestran las pruebas de ejecución de la carga de datos y las de cada uno de estos requerimientos, y se presentan tablas y gráficas que comparan las tomas de los diferentes requerimientos, con el fin de verificar que las complejidades establecidas en el análisis de las secciones 1 y 2 coinciden con la práctica.

Sección 1: Carga de datos – Análisis teórico

Estructura utilizada

Como grupo llegamos a la conclusión de que una estructura del catálogo altamente eficiente en tiempo de búsqueda sería una tabla de hash cuyas llaves sean los años del registro y sus valores sean tablas de hash secundarias que contengan la estructura jerárquica de la información para cada uno de los años (Sectores-Subsectores-Actividades).

En el siguiente esquema (Figura 1) se puede ver una representación visual de nuestra estructura:

Esquema de la estructura propuesta

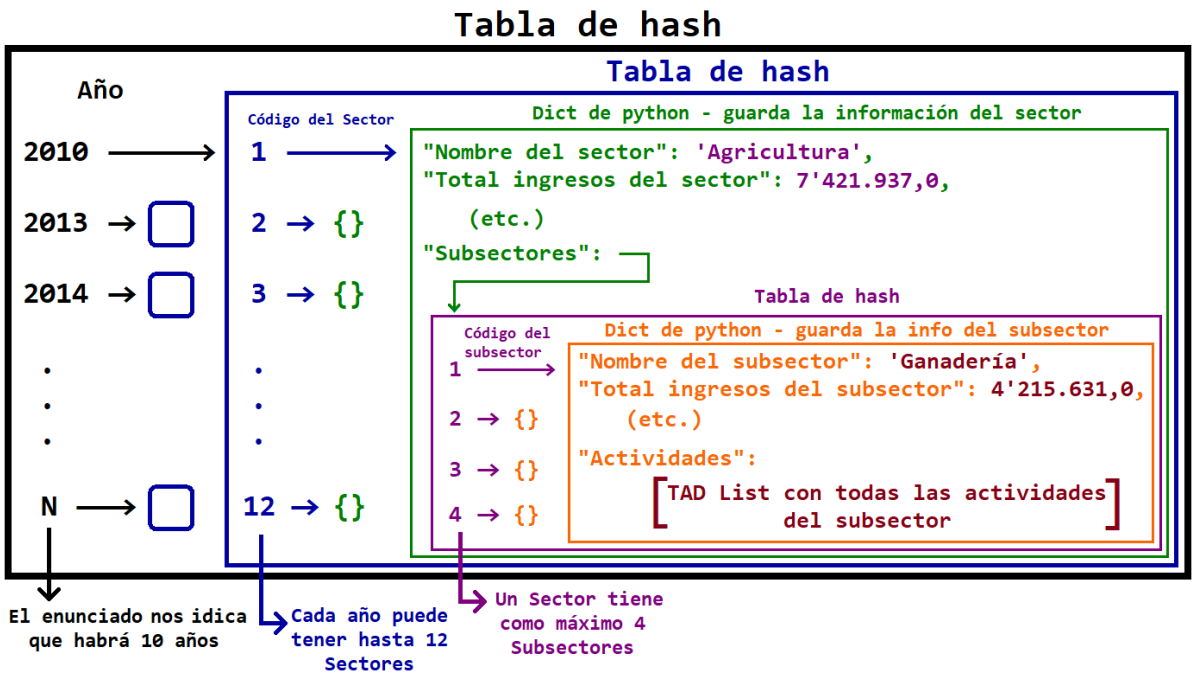


Figura 1

La justificación de esta propuesta de estructura para el catálogo parte de un análisis preciso de las instrucciones de los requerimientos cuyas preguntas guías fueron: ¿Qué nos piden en cada requerimiento?, ¿Qué información necesitamos para todos y, además, para cada uno? Y ¿Cómo vamos a acceder de manera eficiente (principalmente en términos de tiempo) a esa información?

El primer hecho notorio es que todos los requerimientos están filtrados, en primera instancia, por el año del registro. Por este motivo, la estructura principal de nuestro catálogo se trata de una tabla de hash cuyas llaves son cada uno de los años del archivo, ya que esto nos permite acceder a la información perteneciente a un año específico en un tiempo de O(1).

Por la naturaleza de los datos, el segundo gran filtro de la información es el Sector Económico. Por esta razón, continuando con el objetivo de garantizar el acceso a la información con una complejidad temporal O(1), definimos el valor asociado a cada año como una tabla de hash secundaria cuyas llaves serán los códigos de cada uno de los Sectores (máximo 12 por año) y sus valores serán la información específica del Sector correspondiente.

Por temas de simplicidad de código para la manipulación y edición de la información del Sector, y haciendo uso de la flexibilidad que nos brindó el equipo de EDA, decidimos implementar un diccionario nativo de Python para contener la información relativa a cada Sector específico. Más adelante explicaremos que del mismo modo utilizamos esta estructura para almacenar la información de cada Subsector. Esta decisión se debe a que la información de cada Sector y Subsector debe construirse y actualizarse a medida que se lee la base de datos, pues es información que no se encuentra de manera explícita en una ubicación puntual del archivo, sino de manera implícita en cada una de las Actividades Económicas contenidas en el registro. Es decir, cada vez que se lee una línea del archivo, que corresponde a una Actividad particular, no sólo deben almacenarse los datos de dicha Actividad, sino que también deben actualizarse los valores para el Sector y Subsector en los que se clasifica y, en caso de que sea la primera Actividad registrada en estos grupos, debe crearse la estructura jerárquica correspondiente y articularse a la estructura del catálogo.

Todas estas operaciones que tienen que ver con la creación y actualización de conjuntos de información son mucho más sencillas de realizar por medio de los diccionarios de Python, ya que estos nos ofrecen la posibilidad de actualizar directamente estos valores con el operador +=, mientras que si lo hiciéramos de otra forma (con una tabla de hash del curso de EDA, por ejemplo) tendríamos que: acceder al valor, almacenarlo en otra variable, sumar (o restar) el nuevo valor que se está leyendo, y luego volver a asignar el valor editado a su llave correspondiente.

Retomando la explicación de nuestra estructura, el diccionario de un Sector no sólo contiene los valores totales de cada impuesto para el Sector; también tiene una llave denominada “Subsectores” que, como su nombre lo indica, nos permitirá adentrarnos un paso más en la estructura para llegar a la información de los Subsectores Económicos clasificados en el Sector en cuestión (máximo 4 Subsectores por cada Sector). El valor asociado a la llave “Subsectores” será una última tabla de hash que funciona de la misma forma que la tabla de los Sectores para un año: las llaves son los códigos de cada Subsector y los valores son su respectiva información. Como lo justificamos previamente, la información del Subsector será un diccionario nativo de Python donde se almacenen los valores totales de impuestos.

Una vez obtenido el diccionario de un Subsector, sólo nos queda acceder a la información más básica de este proyecto y que es la que sustenta toda la estructura: las Actividades Económicas. Estas se

encontrarán en la llave “Actividades” de cada Subsector, ya filtradas desde la lectura del archivo por el año, el Sector y el Subsector. Esta última estructura que contiene a las Actividades, la hemos definido como un TAD – Array List, ya que de una u otra forma, con las herramientas que tenemos y conocemos hasta ahora, es seguro que tendremos que llegar a un punto de iteración sobre todo el grupo de Actividades para poder compararlas y así encontrar aquellas que cumplan con las condiciones específicas de cada requerimiento.

Podría parecer incoherente o un esfuerzo innecesario el hecho de crear toda una estructura diseñada con el objetivo de alcanzar una complejidad temporal de acceso a la información O(1) para que luego la base de la pirámide sea una estructura cuya complejidad de búsqueda y comparación es O(n). Sin embargo, existe un dato que justifica por qué vale la pena y por qué tiene sentido hacerlo de esta manera:

Cada Subsector de un año tiene en promedio 23 Actividades. Esto es relevante en el sentido de que nuestra estructura garantiza que las Actividades serán clasificadas correctamente desde la carga de datos en sus respectivos años, Sectores y Subsectores, de modo que las listas asociadas a los Subsectores tendrán un tamaño promedio de 23 posiciones. Esto significa que, en el peor caso, si debemos recorrer todas las actividades de un Subsector y un año específicos, la iteración tendrá complejidad temporal de O(n=23). Como podemos estar seguros de que la cantidad de actividades no va a variar, esta complejidad que en teoría es O(n), es en realidad una complejidad de O(23) = O(constante) y, por lo tanto, podemos afirmar teóricamente que O(constante) = O(1).

En conclusión, nuestra propuesta para la estructura del catálogo garantiza en teoría una complejidad temporal de acceso a cualquier nivel de la información de O(1), alcanzando la máxima eficiencia posible.

Beneficios de la estructura utilizada

Es bastante evidente que la estructura que utilizamos para guardar los datos es un tanto enrevesada, pero estas complicaciones tienen una razón de ser muy clara cuando llegamos a la solución de cada uno de los requerimientos. En muchos de estos se nos pide analizar la información de un Sector y/o Subsector en específico, y en el caso de que no hayamos filtrado la información por sectores o subsectores, nos encontraríamos en una situación bastante complicada: sería necesario recorrer todas las actividades para encontrar cuáles forman o no parte de un sector o subsector en específico, cosa que tomaría mucho más tiempo.

Esta es, entonces, la razón de ser de las aparentes complicaciones que tomamos en la carga de datos: hicimos un catálogo con una estructura compleja para poder acceder a la información que nos interesa de forma muchísimo más eficiente. En otras palabras, sacrificamos un poco de complejidad temporal y espacial en la carga con la finalidad de minimizarla en cada uno de los requerimientos.

Análisis de complejidad espacial de la carga

Hacer este análisis es más sencillo si lo hacemos de lo más grande a lo más pequeño. Vemos que la tabla de hash exterior tiene 10 parejas llave valor. Como la complejidad espacial de una tabla de Hash de m elementos es O(m), concluimos que la complejidad de esta tabla es O(10) = O(1).

En el siguiente nivel vemos que hay una tabla de hash de a lo sumo 12 valores, por lo que la complejidad espacial es O(12). Como hay 10 de estas, la complejidad espacial de todas consideradas en conjunto es O(10*12) = O(120) = O(1).

El siguiente nivel es un diccionario que tiene k+1 parejas llave valor, donde k es el número de metadatos que se quiere guardar del sector. Luego, la complejidad de cada uno de estos diccionarios es O(k+1). Como hay 120 diccionarios de estos, la complejidad espacial de estos diccionarios considerados en conjunto es O(120(k+1)). Como k es una constante, la complejidad de todas estas estructuras es O(constante) = O(1).

El siguiente nivel es una tabla de Hash de a lo sumo 4 parejas, por lo que cada uno tiene una complejidad de O(4). Como hay tantas tablas de estas como sectores, hay a lo sumo 120 tablas en este nivel. Luego, la complejidad espacial de todas consideradas en conjunto es: O(120*4) = O(480) = O(constante) = O(1).

El siguiente nivel es un diccionario que tiene p+1 parejas llave valor, donde p es el número de metadatos que se quiere guardar del subsector. Luego, la complejidad de cada uno de estos diccionarios es O(p+1). Como hay 480 diccionarios de estos, la complejidad espacial de estos diccionarios considerados en conjunto es O(480(p+1)). Como p es una constante, la complejidad de todas estas estructuras es O(constante) = O(1).

El último nivel es un array-list donde se guardarán todas las actividades de ese subsector. Como hay en total n actividades y cada una de estas actividades va a entrar en exactamente una de estas listas, la complejidad espacial de todas estas listas pensadas en conjunto es O(n).

Por tanto, la complejidad espacial total de la carga es:

$$O(5 \cdot 1) + O(n) = O(1) + O(n) = O(n + 1) = O(n).$$

Sección 2: Requerimientos – Análisis teórico

Requerimientos básicos (1 y 2)

Descripción

Estos requerimientos se encargan de encontrar la actividad económica con mayor total saldo a pagar (1) y mayor rotal saldo a favor (2) para un año específico. Estas diferentes opciones son asignadas a una variable llamada monto la cual cambiará según el requerimiento:

1) Se accede al catálogo con la llave año, para obtener la tabla de hash de los sectores de dicho año.
2) Se accede en la tabla de hash a la llave del sector económico que se recibió como parámetro y luego se accede a la tabla de hash que contiene todos los subsectores de dicho sector.
3) Se inicializan dos variables para la solución del requerimiento.
4) Se recorren todas las llaves de la tabla de hash que se acaba de acceder para luego recorrer para cada subsector sobre todas las actividades del subsector, actualizando en cada iteración cuál es la actividad que más aportó.

Entrada	1. Estructura de datos principal del modelo (el catálogo) 2. Y el año de interés.
Salidas	Actividad económica con mayor 'Total saldo a pagar' (Req 1) o 'Total saldo a favor' (Req 2) para el año ingresado.
Implementado (Sí/No)	Sí. Se hizo grupalmente.

Complejidad

Pasos	**Complejidad**
Paso 1	O(1)
Paso 2	O(1)
Paso 3	O(1)
Paso 4	O(n)
TOTAL	***O(n)***

Justificación

El primer paso y el segundo, al consistir en accesos en tablas de hash o diccionarios, tiene una complejidad temporal de O(1).

El tercer paso, al consistir en asignación de variables, también tiene una complejidad temporal de O(1).

El cuarto paso consiste en un recorrido dentro de otro. El recorrido externo corresponde a un ciclo sobre los subsectores de un sector, y como en promedio cada sector tiene 3 subsectores (esto se puede ver analizando los archivos csv con los datos). El recorrido interno consiste en iterar sobre las actividades del subsector que se está recorriendo, cosa que tiene una complejidad de O(act) donde act es el número de actividades en un subsector promedio. Como el número de actividades de un subsector en un año crece linealmente con el número de datos de la muestra, podemos concluir que la complejidad de este recorrido interno es O(n). Luego, la complejidad de todo el paso es O(3n)=O(n).

Sumando las complejidades nos queda: $O(1) + O(1) + O(1) + O(n) = O(n + 1) = O(n)$.

Requerimientos individuales (3, 4 y 5)

Descripción

Este requerimiento se encarga de encontrar en un año especifico, el subsector económico con el menor total de retenciones (3), con los mayores costos y gastos (4) o con los mayores descuentos tributarios (5). Estas diferentes opciones son asignadas a una variable llamada monto la cual cambiará según el requerimiento:

1) Se inicializan las variables que dependen del requerimiento en None y se le asignan los valores según corresponden. Se asignan diferentes funciones de comparación para cada uno de los requerimientos. Igualmente, se inicializan las variables necesarias para identificar el mayor subsector.
2) Se obtiene la tabla de hash que contiene los sectores del año y se empieza a recorrer. Este ciclo hará máximo 12 iteraciones, ya que únicamente existen 12 sectores.
3) Se obtiene la tabla de hash de los subsectores y esta se empieza a recorrer. Este ciclo tendrá máximo 4 iteraciones, ya que solo puede haber 4 subsectores por sector.
4) Se compara el monto del subsector con el mayor monto que se haya almacenado, si es mayor se actualizan los valores necesarios. Finalmente, obtenemos la variable mayor al subsector de respuesta.
5) Al obtener el subsector, editamos las columnas necesarias por requerimiento, añadimos la columna “monto” a esta.
6) Utilizamos una función sort para buscar los primeros tres y últimos tres según el monto. De tal modo que, únicamente se organizan que los primeros tres queden organizados y últimos tres queden organizados. Esto se hizo mediante una función sort que consistía en pararse en el primer elemento, recorrer toda la lista y encontrar el mínimo. Al encontrarlo este lo intercambiaba con la función exchange, y repetía el mismo proceso para los primeros 3 y últimos 3 de la lista.

Entrada	1. Estructura de datos principal del modelo (el catálogo) 2. Año 3. El número del requerimiento que la utiliza (3, 4 o 5).
Salidas	Encuentra el subsector económico que tuvo: Req 3: El menor total de ‘Retenciones’ para cada año. Req 4: El mayor total de ‘Costos y gastos nómina’ para cada año. Req 5: El mayor total de ‘Descuentos tributarios’ para cada año.
Implementado (Sí/No)	Sí. Aunque eran individuales, por las similitudes de los requerimientos el algoritmo para solucionarlos tiene la misma estructura. Por esta razón se desarrollaron de forma unificada. Sin embargo, cada uno de los miembros se encargó principalmente de la implementación de su requerimiento: Requerimiento 3: Tomás Iriarte. Requerimiento 4: Juan Esteban Arboleda Requerimiento 5: Verónica Isaza

Complejidad

Pasos	Complejidad
Paso 1	O(1)
Paso 2	O(1)
Paso 3	O(1)
Paso 4	O(1)
Paso 5	O(1)
Paso 6	O(n)
TOTAL	***O(n)***

Justificación

El primer paso, al consistir en asignar variables y funciones de comparación de acuerdo con el requerimiento que se quiera ejecutar, este tiene complejidad O(1). Pues únicamente se está asignando y creando variables.

El segundo paso recorre la tabla de hash que tiene los códigos de los sectores. De modo que, al tener 12 sectores, lo máximo que iterará será 12 veces, por lo que su complejidad será de O(12) lo cual se puede acotar a O(1).

El tercer paso consiste en recorrer la tabla de hash de los subsectores. AL igual que en el anterior paso, este tendrá un límite de iteraciones al tener máximo 4 subsectores por sector. Es por esto que la complejidad será de O(4) que igualmente se puede acotar a O(1). Como este ciclo esta adentro del anterior su complejidad sería de O(12*4) = O(48) = O(1) igualmente.

El cuarto paso compara el monto con el mayor monto que se haya almacenado y se actualiza con base a esto. Luego, se obtiene el subsector mayor y su valor en cuanto al monto. Al ser una comparación, este paso será de O(1) al igual que los otros.

El quinto paso consiste en edita las columnas por el requerimiento y añade la columna “monto”, la longitud de esta lista de columnas será constante y tendrá complejidad de O(1).

El sexto paso, organizaba los primeros 3 y últimos 3 datos de la lista obtenida. Este recorría los elementos de la lista para encontrar los 3 primeros y últimos, para así mejorar su complejidad y obtener una de O(n).

Sumando las complejidades nos queda:

$$O(1) + O(1) + O(1) + O(1) + O(1) + O(n) = O(n + 1) = O(n).$$

Requerimiento 6

Descripción

Este requerimiento se encarga de encontrar la actividad económica con el mayor total de ingresos netos para cada sector económico en un año específico. Se compone de los siguientes pasos:

1) Se accede al catálogo con la llave año, para obtener la tabla de hash de los sectores de dicho año.
2) Se inicializan variables.
3) Se recorre el mapa de sectores que se extrajo en (1). Esto se hace para determinar que sector fue el que más aportó en el año.
4) Se inicializan más variables.
5) Se recorren los subsectores del sector para determinar cuál subsector aportó más y cuál menos.
6) Se editan las columnas del sector y de los subsectores que más y menos aportaron para dar formato.
7) Se recorre la lista de actividades de cada sub para determinar qué actividad aporta más o menos en los dos subsectores.
8) Se organizan listas y una cola para el retorno.

Entrada	Estructuras de datos del modelo, año.
Salidas	La actividad económica con el mayor total de ingresos netos para cada sector económico en un año específico.
Implementado (Sí/No)	Sí. Se hizo grupalmente.

Complejidad

Pasos	**Complejidad**
Paso 1	O(1)
Paso 2	O(1)
Paso 3	O(1)
Paso 4	O(1)
Paso 5	O(1)
Paso 6	O(1)
Paso 7	O(n) en teoría, pero en la práctica casi O(1).
Paso 8	O(1)
TOTAL	***O(n)****

Justificación

El primer paso, al consistir en accesos en un acceso a tablas de hash, tiene una complejidad temporal de O(1).

El segundo paso, al consistir en inicialización de variables, tiene una complejidad temporal de O(1).

El tercer paso consiste en un recorrido sobre los sectores de un año. Como hay 12 sectores, se harán como máximo 12 iteraciones. Luego, la complejidad en el peor de los casos es O(12)=O(1).

El cuarto paso, al consistir en inicialización de variables, tiene una complejidad temporal de O(1).

El quinto paso consiste en un recorrido sobre los subsectores de un sector. Como en un sector hay como máximo 3 subsectores, la complejidad en el peor de los casos es O(3)=O(1).

En el sexto paso se editan ligeramente las columnas del sector que más aportó, del subsector que más aportó y del subsector que menos aportó. Como en cada uno de los 3 casos se agregan 4 columnas, la complejidad de este proceso es O(3*4)=O(12)=O(1).

El séptimo paso consiste en dos recorridos. El primero recorre las actividades del subsector que más aportó y el segundo las actividades del subsector que menos aportó. Como en total hay 10 años, dentro de cada año hay 12 sectores y dentro de cada sector hay en promedio 3 subsectores, podemos decir que hay 10*12*3=360 listas de subsectores, entre las cuales se deben distribuir n actividades económicas. Luego, en promedio estas listas tienen n/360 elementos. Como hay que recorrer dos de estas, en total hay que iterar sobre n/180 actividades. Como el archivo más grande que se está utilizando tiene un número de actividades n=4903, podemos ver que en los casos de prueba presentados, se harán como máximo 27 iteraciones, cosa que en la práctica es tan bueno como O(1). Luego, aunque en teoría la complejidad de este paso es O(n), en la práctica se comporta de forma casi idéntica a O(1).

En el octavo paso se crean 3 listas, lo cual tiene complejidad O(1), agregándole a cada una solo un elemento, que también es O(1). Luego, se crea una cola a la cual se le agrega cada una de las listas antes mencionadas. Todos estos 3 subpasos son O(1), por lo que la complejidad de todo es O(3*1)=O(3)=O(1).

Sumando todas las complejidades nos queda O(1)+O(1)+ O(1)+O(1)+ O(1)+O(1)+O(n)+O(1)=O(n). Sin embargo, en la práctica, debido a que el paso 6 en la práctica se comporta casi como O(1), podemos decir que la complejidad del algoritmo en la práctica sería de 8*O(1)=O(1).

Requerimiento 7

Descripción

Este requerimiento se encarga de listar el TOP (N) de las actividades económicas con el menor total de costos y gastos para un año especifico y subsector especifico. Se compone de los siguientes pasos:

1) Se asigna None a una variable llamada lista_return y se empieza a recorrer la tabla de hash de los sectores, la cual tendrá máximo 12 iteraciones.
2) Se verifica si el código del subsector ingresado por el usuario está en el diccionario de subsectores. Si este se encuentra se copia la información del subsector y se asigna la lista de todas las actividades del subsector a la variable lista_return. Si el código no se encontraba, lista_return continúa siendo None y retorna esto.
3) Se utiliza una función para organizar las actividades según el menor total de costos y gastos y se para el ciclo.
4) Se verifica si el tamaño de la lista es mayor a la cantidad que pide el usuario, esta se reduce a la cantidad que es pedida con la función lt.subtlist().

Entrada	Estructuras de datos del modelo, año inicial, año final, N.
Salidas	TOP (N) de las actividades económicas con el menor total de costos y gastos para un subsector y un año especifico.
Implementado (Sí/No)	Sí. Se hizo grupalmente.

Complejidad

Pasos	**Complejidad**
Paso 1	O(1)
Paso 2	O(1)
Paso 3	O(n)
Paso 4	O(1)
TOTAL	***O(n)***

Justificación

El primer paso, consiste en asignar una variable a None y en recorrer una tabla de hash con máximo 12 iteraciones. La asignación será de O(1) y el recorrido de la table será de O(12) luego este paso en general es de O(1).

El segundo paso, verifica si el código del subsector ingresado por el usuario está en el diccionario de subsectores. Buscar si está el elemento en un diccionario es O(1), al igual que copiar la información y asignar una variable a list_return. De modo que todo esto es O(1).

El tercer paso, utiliza la función anteriormente mencionada para organizar los datos. En esta, se recorre la lista y se van intercambiando los valores para organizarlos para la cantidad necesaria. De modo que, esto es O(n), pues recorre toda la lista para organizar los primeros valores deseados.

El cuarto paso consiste en reducir la lista a la cantidad que desee el usuario. Esto se hace creando una sublista y añadiendo los elementos hasta que se complete la cantidad que se quiere, por lo que esto tiene una complejidad de O(1).

Sumando todas las complejidades nos queda: $O(1) + O(1) + O(n) + O(1) = O(n + 1) = O(n)$.

Requerimiento 8 (bono)

Descripción

Este requerimiento se encarga de listar un TOP N de Actividades Económicas dentro de cada Subsector de un año específico. Es decir, para cada Subsector del año requerido se encuentra el TOP N y luego se muestran sus resultados. Este top se define el según monto de impuestos a cargo de cada actividad.

La solución de este requerimiento se compone de los siguientes pasos:

1) El primer paso es crear una lista vacía en la que se van a almacenar los Subsectores del año, con el objetivo de poder mostrar su información en pantalla posteriormente.
2) Teniendo el año correspondiente como llave, se accede a la tabla de Sectores asociada al año.
3) Para poder recorrer cada uno de los Subsectores, deben recorrerse así mismo todas las llaves de la tabla (22 Subsectores en total repartidos entre 12 Sectores) por lo que tenemos en este caso una complejidad temporal de O(22) = O(1).
4) Ahora bien, a medida que se recorren los Subsectores del año, se hace una operación sobre la lista de Actividades del sector que consiste en buscar 3 veces la menor y 3 veces la mayor. De esta forma, obtendremos las primeras y últimas 3 actividades de cada sector. Esto tiene una complejidad de O(6*n) = O(n).
5) Una vez compuesta la lista de Subsectores del año, se ejecuta sobre ella un ordenamiento Merge Sort. Este paso es en teoría O(nlogn), pero como el número de subsectores es constante (n=22), la complejidad de este ordenamiento es en realidad O(22*log(22)) = O(98) = O(constante) = O(1).
6) Por último, debido a las restricciones del enunciado, verificamos el tamaño de la lista de Subsectores y, si esta es mayor a 12, se toman sólo los primeros y últimos 3. Este paso tiene una complejidad de O(n).

Entrada	El catálogo, el año de interés y el número N del top deseado.
Salidas	TOP (N) de las Actividades Económicas de cada Subsector con los mayores totales de impuestos a cargo para el año solicitado.
Implementado (Sí/No)	Sí. Se hizo grupalmente.

Complejidad

Pasos	**Complejidad**
Paso 1	O(1)
Paso 2	O(1)
Paso 3	O(1)
Paso 4	O(n)
Paso 5	O(1)
Paso 6	O(n)
TOTAL	***O(n)***

Justificación

La implementación de este requerimiento garantiza la menor complejidad temporal posible utilizando las estructuras y herramientas que hemos visto en el curso hasta ahora: O(n) para comparación y búsqueda de máximos y mínimos elementos.

La estructura y organización del catálogo nos ayuda a reducir ordenamientos y búsquedas sobre grandes volúmenes de datos para así pasar de O(nlogn) con un número máximo de n= 495 a grupos más pequeños con una complejidad de O(n) con un número promedio de n=23.

Sección 3: Pruebas de ejecución – Tablas y graficas

Máquinas utilizadas

Las máquinas en las que se realizaron las pruebas de ejecución tienen las siguientes especificaciones:

	Máquina 1	Máquina 2	Máquina 3
Procesadores	AMD Ryzen 7 5700U with Radeon Graphics 1.80 GHz	AMD A8-5550M APU with Radeon(tm) HD Graphics 2.10 GHz	Intel(R) Core (TM) i5-10210U CPU @ 1.60GHz 2.11 GHz
Memoria RAM (GB)	16.0 GB	16 GB	8.00 GB
Sistema Operativo	Windows 11 Home Single Language	Windows 10pro 64 bits	Windows 11 Home – 64 bits

Tabla 1. Especificaciones de las máquinas para ejecutar las pruebas de rendimiento.

Pruebas de la carga de datos

Descripción de las pruebas realizadas

Debido a que nuestro diseño del catálogo surgió de un análisis cuyo objetivo era alcanzar la mayor eficiencia temporal posible, las pruebas realizadas en la carga de datos se llevaron a cabo con mediciones no sólo de la complejidad espacial, sino también de la complejidad temporal del proceso de construcción de la estructura, de modo que los resultados pudiesen confirmar o rechazar nuestro análisis inicial. Teniendo en cuenta que el análisis de complejidad espacial realizado en el Laboratorio 7 evidenció que en la práctica no hay una diferencia relevante entre los dos mecanismos de colisión para la complejidad espacial del Reto 2, decidimos realizar las pruebas de la carga sobre la estructura Separate Chaining.

Las pruebas de memoria consisten en medir el cambio del espacio utilizado en memoria para almacenar los datos en función del tamaño de la muestra.

En cuanto a las pruebas de tiempo, estas consisten en medir el tiempo de procesamiento (o tiempo de CPU) que consume el conjunto de operaciones que se quieren evaluar. El tiempo de CPU es diferente al tiempo real o tiempo de ejecución, en el sentido de que es mucho más preciso y acertado para medir la complejidad temporal de un bloque de código.

En este proyecto, logramos implementar una automatización de las pruebas de tiempo, de modo que pudiésemos realizar más de una medición de forma mucho más rápida y promediando los datos obtenidos entre un número de pruebas lo suficientemente grande para ser confiable.

Tabla de resultados – Pruebas de memoria

Tamaño de muestra (#)	Carga de datos – Memoria (kB)		
Máquinas	**M1**	**M2**	**M3**
49	289,5	299,4	305,63
245	646,98	784,69	789,22
490	1003,46	1209,21	1213,82
980	1594,14	1899,14	1903,67
1470	2137,35	2531,84	2536,37
2451	3186,85	3753,80	3757,56
3922	4744,76	5566,71	5570,99
4903	5778,08	6769,75	6774,28

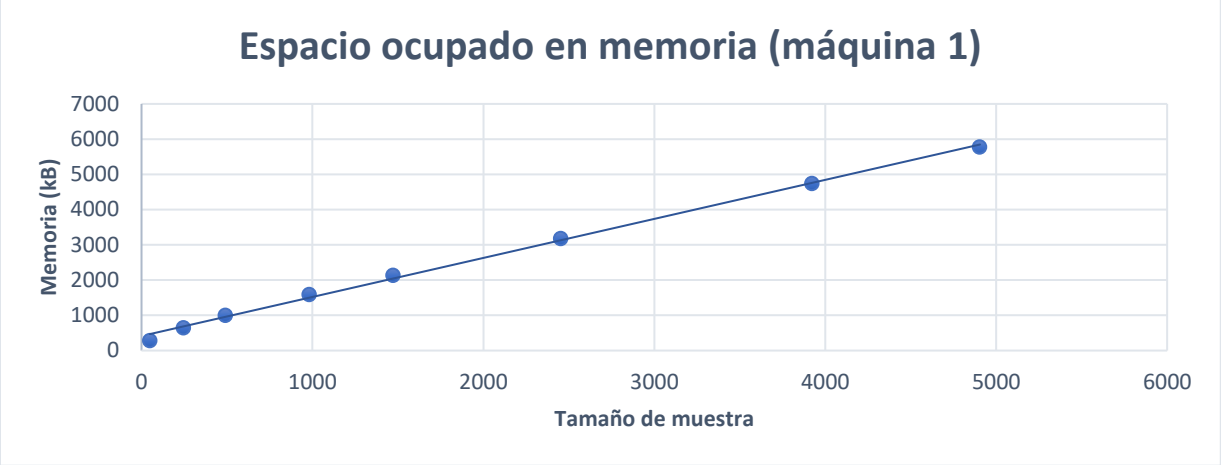
Tabla 2. Tamaño de muestra (#) vs memoria ocupada por el catálogo (kB).

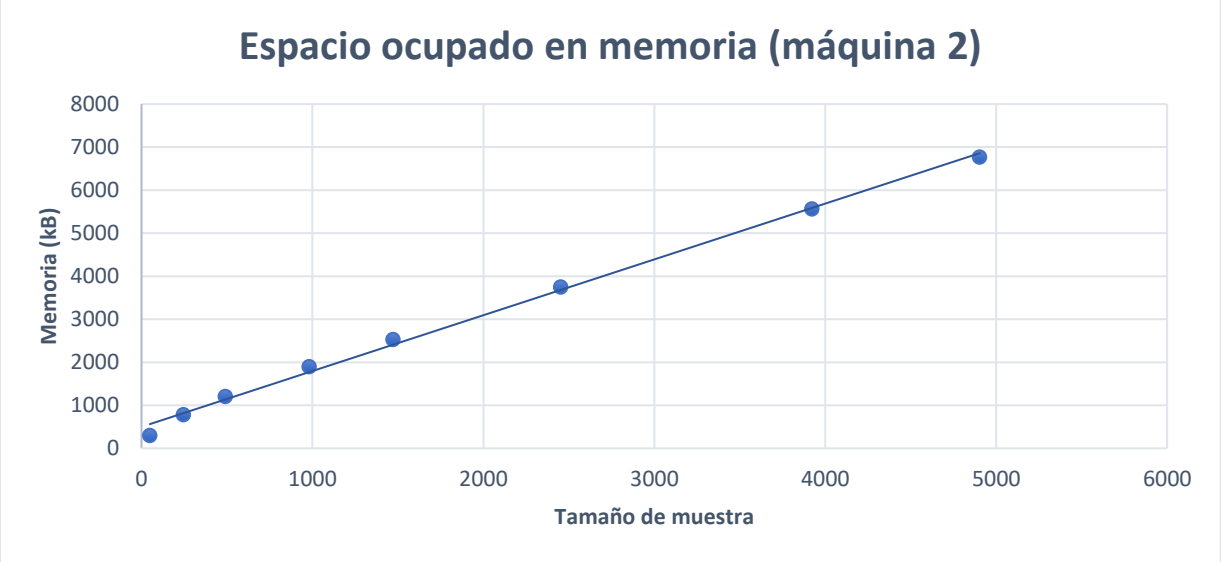
Tabla de resultados – Pruebas de tiempo

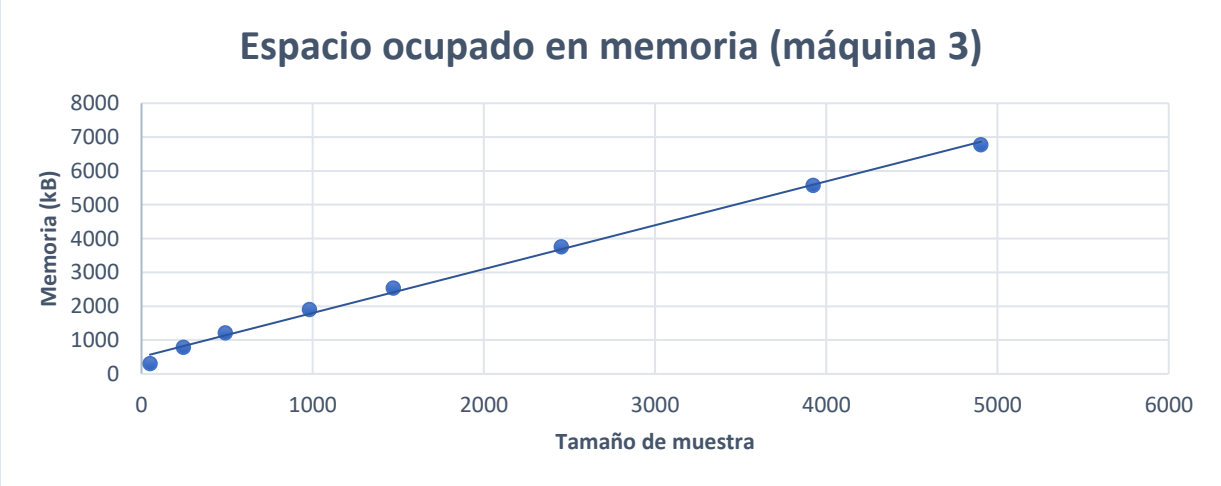
Tamaño de muestra (#)	Máquinas					
	M1		M2		M3	
	Carga (ms)	REQS (ms)	Carga (ms)	REQS (ms)	Carga (ms)	REQS (ms)
49	2,46	0,37	36,24	3,56	6,54	0,50
245	9,68	0,45	109,18	3,75	20,64	0,60
490	18,00	0,49	181,67	4,42	34,89	0,65
980	34,46	0,54	308,41	4,70	57,82	0,73
1470	50,28	0,61	417,53	5,02	80,75	0,80
2451	83,7	0,80	666,16	5,43	129,61	1,07
3922	133,41	0,80	1034,30	6,01	209,93	1,07
4903	166,89	0,89	1254,82	7,10	258,92	1,17

Tabla 3. Tamaño de muestra (#) vs tiempo de carga y tiempo promedio de los requerimientos (ms).

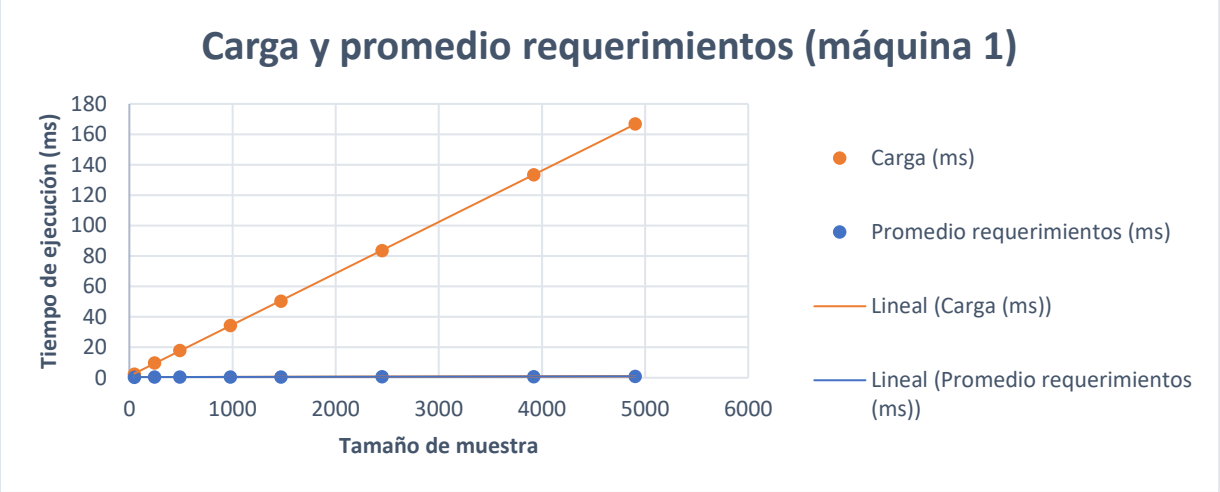
Gráficas – Pruebas de memoria

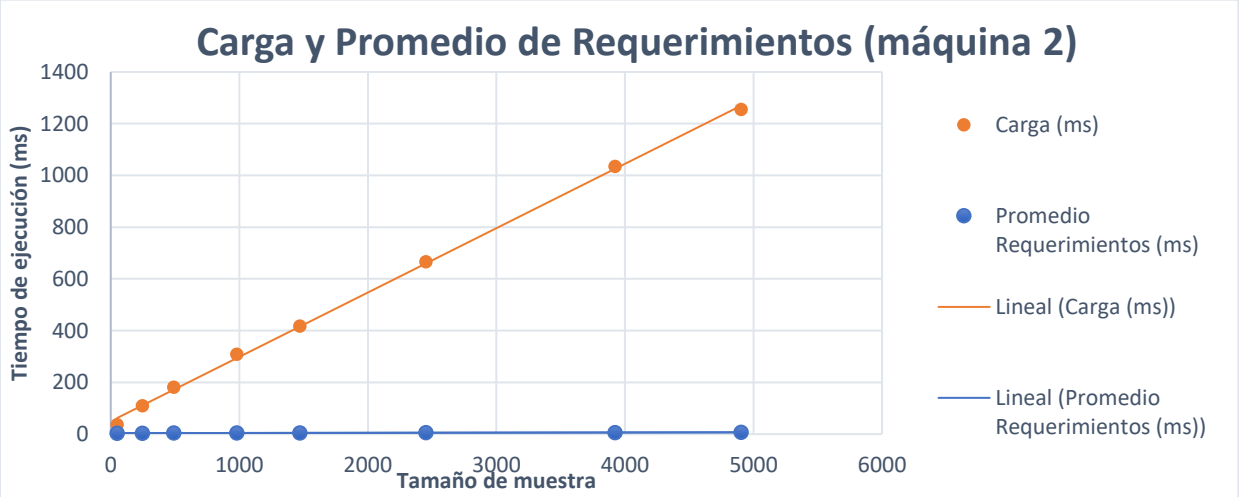


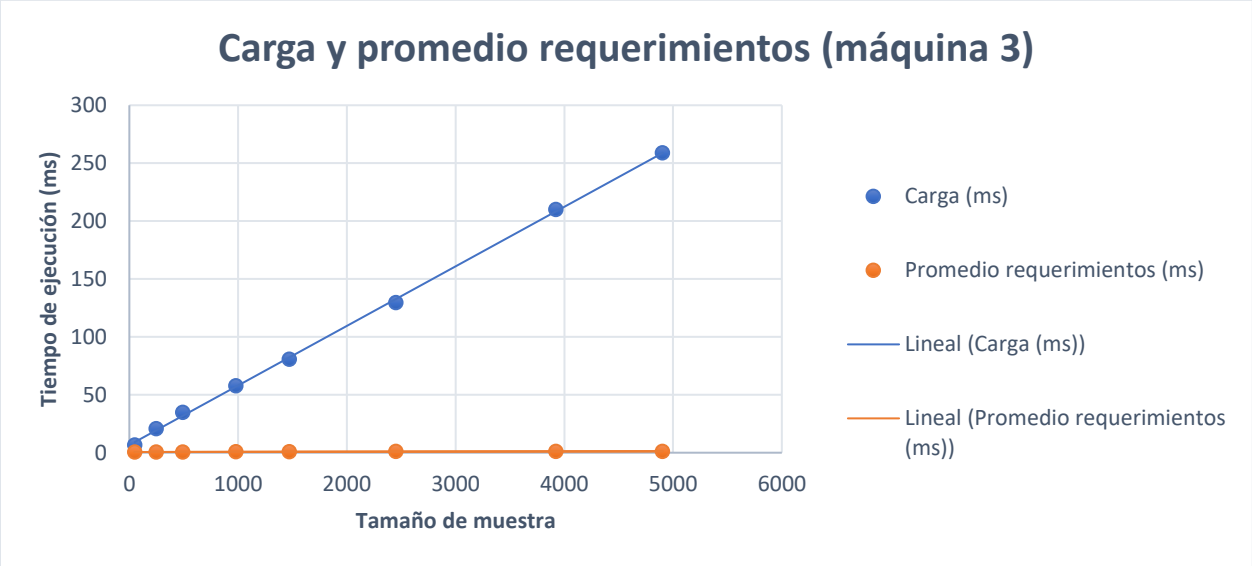




Gráficas – Pruebas de tiempo







Análisis de resultados – Memoria en la Carga de datos

En las gráficas se puede leer claramente cómo el espacio que ocupa el catálogo en memoria se comporta de forma lineal con respecto al tamaño de la muestra. Esto es lo que se esperaba inicialmente porque, como se mencionó en el análisis espacial de la carga, el catálogo tiene una complejidad espacial de O(n). Por tanto, el valor obtenido en las pruebas coincide con el teórico.

Análisis de resultados – Tiempo en la carga de datos

Los resultados arrojados por las pruebas de tiempo nos permiten concluir que nuestro análisis inicial, así como el diseño de la estructura del catálogo, fueron completamente acertados. Como era de esperar, la lectura de los datos se comporta con una complejidad temporal de O(n), de la misma forma que la complejidad espacial. Sin embargo, si nos enfocamos ahora en la complejidad temporal promedio de los requerimientos, podemos observar cómo el comportamiento es casi totalmente igual a una constante, un resultado asombroso que confirmó el éxito total de nuestra estrategia.

Otro aspecto que vale la pena resaltar es el orden de magnitud entre el tiempo de la carga de datos y el tiempo promedio de los requerimientos. Para tomar el ejemplo con la máquina menos eficiente (máquina 2), podemos observar que mientras la carga de datos alcanza un tiempo máximo alrededor de 1.250,0 ms, el tiempo promedio máximo de los requerimientos (en el mismo tamaño de muestra) es de 7,0 ms. Esta proporción es simplemente aplastante, y nos confirma de manera contundente que el esfuerzo por construir una estructura un poco más compleja en términos de organización de subconjuntos de información valió la pena completamente.

Además, y como última observación, también vale la pena recordar las unidades de tiempo con las que estamos haciendo esta medición. Estamos hablando de que la máquina menos eficiente se demora un máximo de 1.250 ms, es decir 1,25 segundos en leer la información y la vez construir la estructura del catálogo. Esto es algo que nos pareció realmente asombroso.

Pruebas requerimientos básicos (1 y 2)

Descripción de las pruebas realizadas

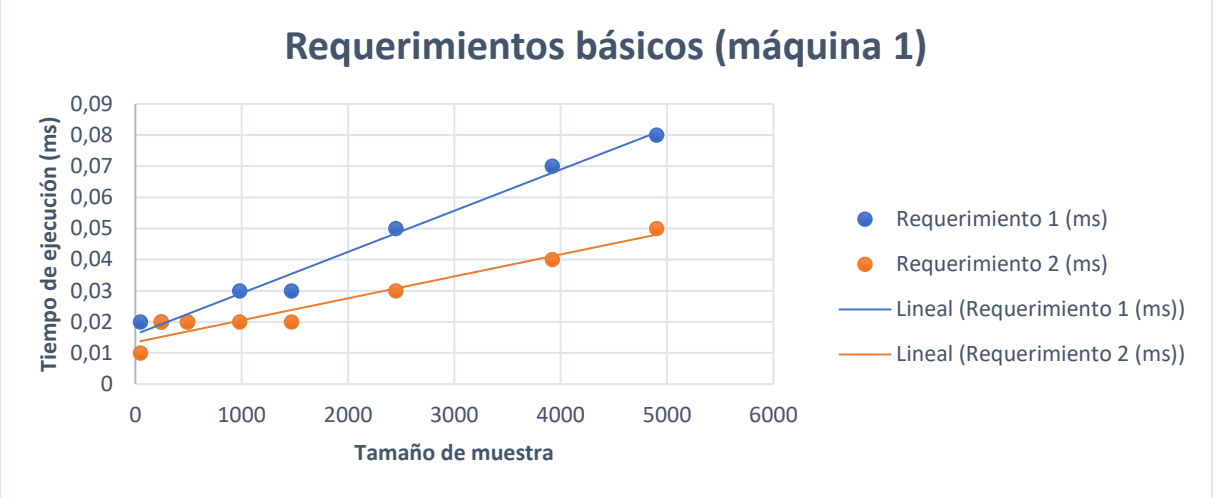
Las pruebas realizadas para estos requerimientos son solo de complejidad temporal. Como estos requerimientos reciben como parámetro un año en específico, lo que se decidió hacer para las pruebas es fijar un año (2021). Además, al recibir también sector como parámetro se fijó un sector en específico (3). Luego, se midió cómo cambiaba el tiempo de ejecución del requerimiento cuando se cambiaba el tamaño de la muestra.

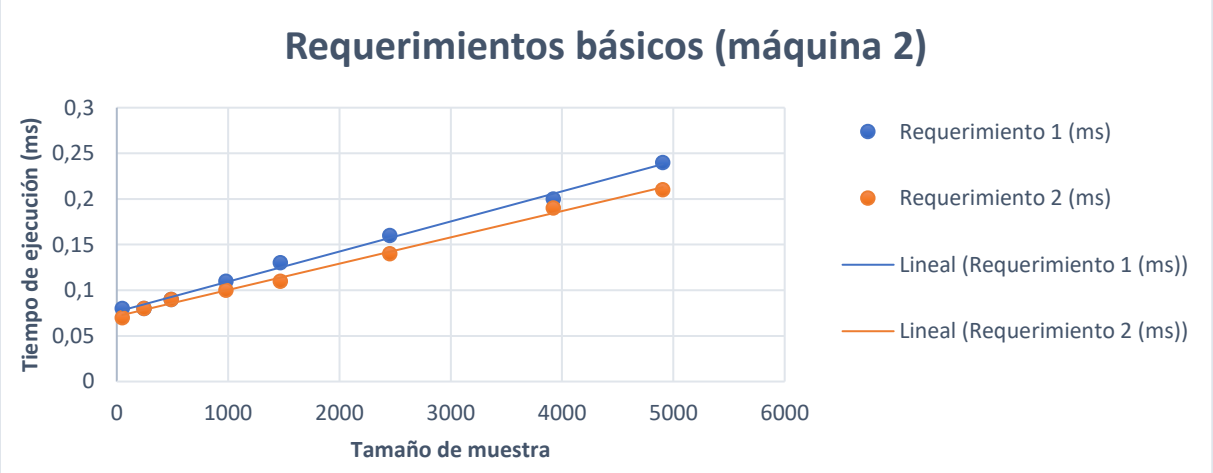
Tablas de datos

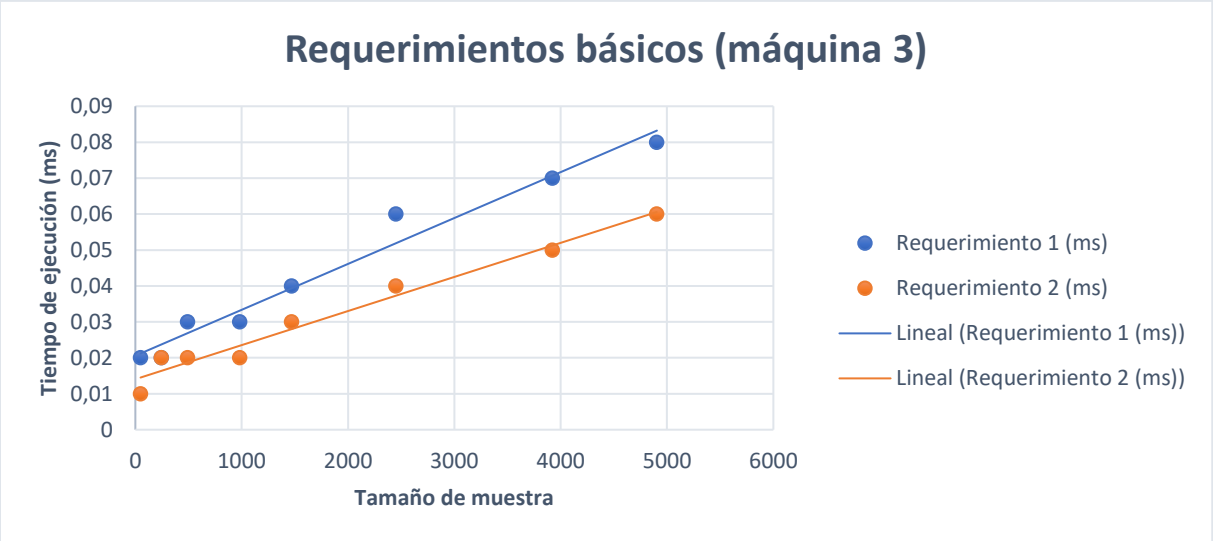
Tamaño de muestra (#)	**Requerimiento 1 – Tiempo (ms)**			**Requerimiento 2 – Tiempo (ms)**		
Máquinas	**M1**	**M2**	**M3**	**M1**	**M2**	**M3**
49	0,02	0,08	0,02	0,01	0,07	0,01
245	0,02	0,08	0,02	0,02	0,08	0,02
490	0,02	0,09	0,03	0,02	0,09	0,02
980	0,03	0,11	0,03	0,02	0,10	0,02
1470	0,03	0,13	0,04	0,02	0,11	0,03
2451	0,05	0,16	0,06	0,03	0,14	0,04
3922	0,07	0,20	0,07	0,04	0,19	0,05
4903	0,08	0,24	0,08	0,05	0,21	0,06

Tabla 4. Tamaño de muestra (#) vs tiempo de ejecución del requerimiento (ms).

Gráficas


Requerimientos básicos (máquina 1)
Tiempo de ejecución (ms)
0,09
0,08
0,07
0,06
0,05
0,04
0,03
0,02
0,01
0
0
1000
2000
3000
4000
5000
6000
Tamaño de muestra
Requerimiento 1 (ms)
Requerimiento 2 (ms)
Lineal (Requerimiento 1 (ms))
Lineal (Requerimiento 2 (ms))


Requerimientos básicos (máquina 2)
Tiempo de ejecución (ms)
0,3
0,25
0,2
0,15
0,1
0,05
0
0
1000
2000
3000
4000
5000
6000
Tamaño de muestra
Requerimiento 1 (ms)
Requerimiento 2 (ms)
Lineal (Requerimiento 1 (ms))
Lineal (Requerimiento 2 (ms))


Requerimientos básicos (máquina 3)
Tiempo de ejecución (ms)
0,09
0,08
0,07
0,06
0,05
0,04
0,03
0,02
0,01
0
0
1000
2000
3000
4000
5000
6000
Tamaño de muestra
Requerimiento 1 (ms)
Requerimiento 2 (ms)
Lineal (Requerimiento 1 (ms))
Lineal (Requerimiento 2 (ms))

Análisis de resultados – Requerimientos básicos

En las gráficas se puede ver que el tiempo de ejecución de los requerimientos 1 y 2 se comporta de forma lineal con respecto al tamaño de la muestra. Esto es lo que se esperaba inicialmente porque, como se dijo en el análisis teórico, estos requerimientos tienen una complejidad temporal de O(n). Por tanto, el valor obtenido en las pruebas coincide con el teórico.

Pruebas requerimientos individuales (3, 4 y 5)

Descripción de las pruebas realizadas

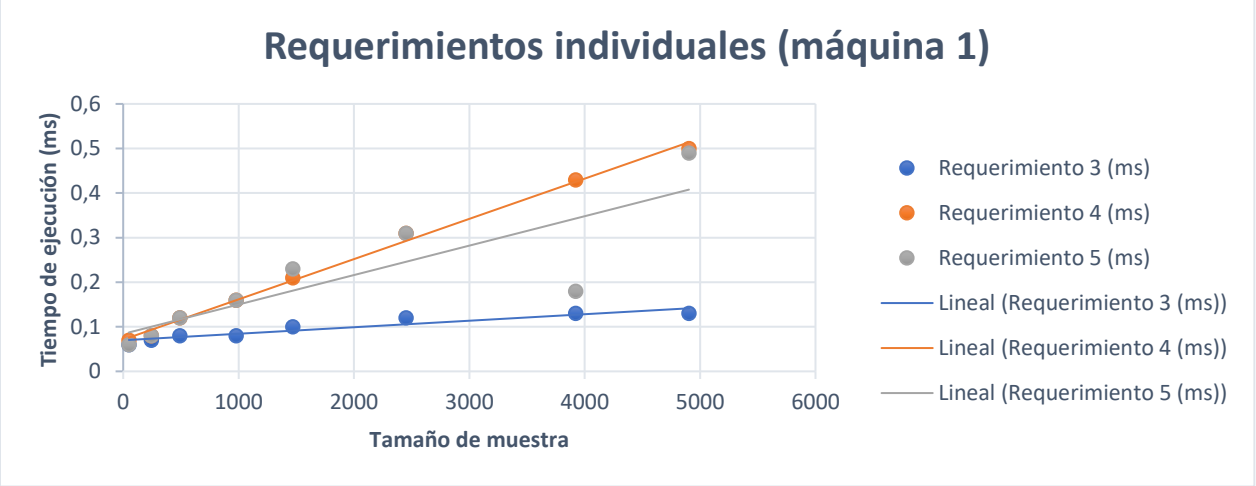
Las pruebas realizadas para estos requerimientos son solo de complejidad temporal. Como estos requerimientos reciben como parámetro un año en específico, lo que se decidió hacer para las pruebas es fijar un año (2021). Luego, se midió cómo cambiaba el tiempo de ejecución del requerimiento cuando se cambiaba el tamaño de la muestra.

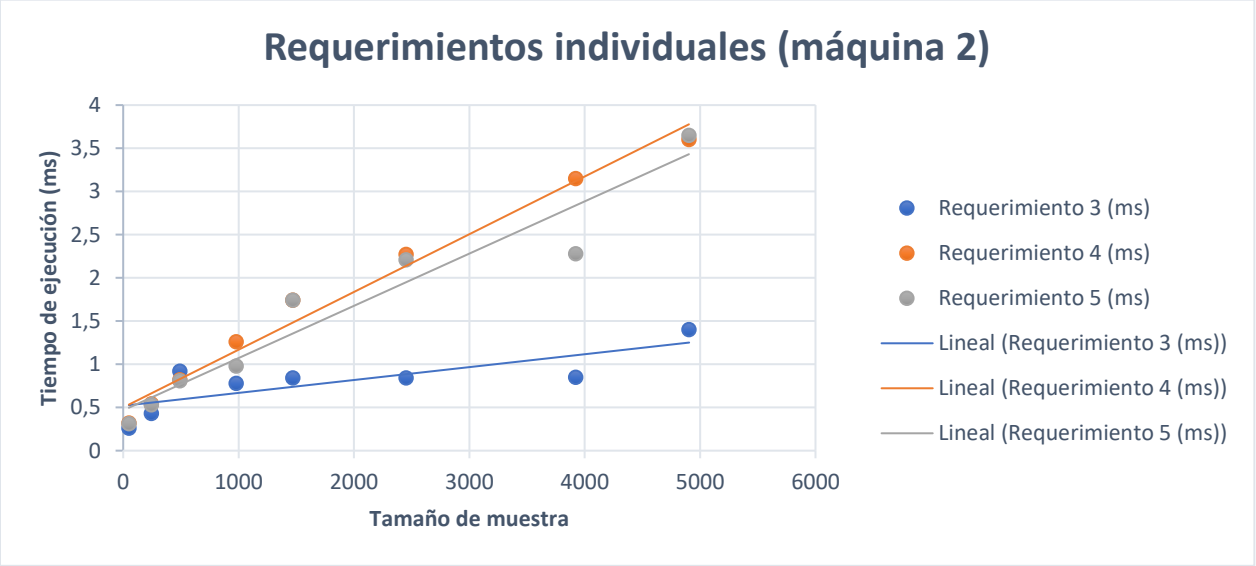
Tablas de datos

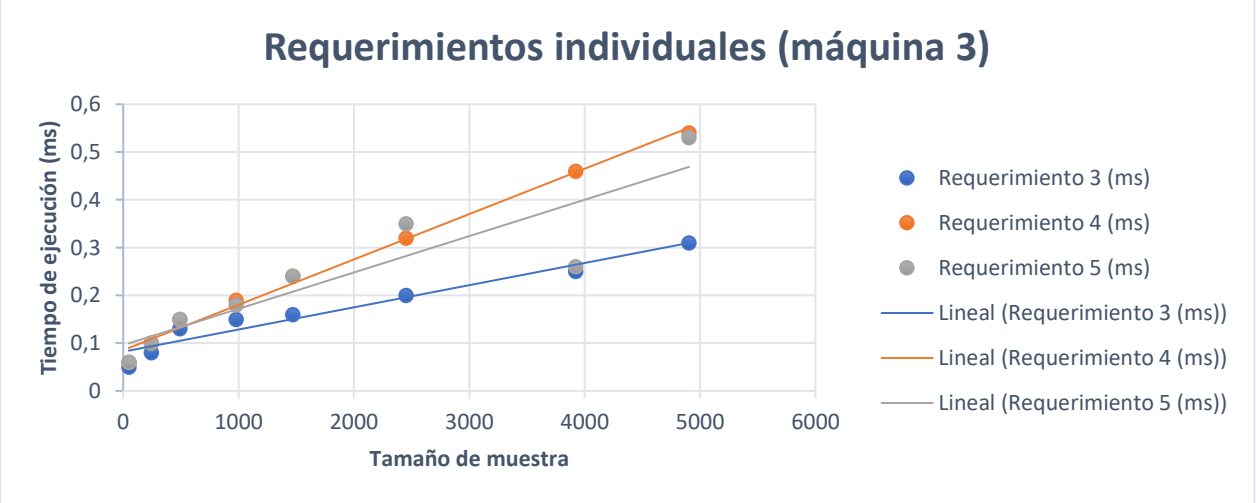
Tamaño de la muestra (#)	**Requerimiento 3 – Tiempo (ms)**			**Requerimiento 4 – Tiempo (ms)**			**Requerimiento 5 – Tiempo (ms)**		
Máquinas	**M1**	**M2**	**M3**	**M1**	**M2**	**M3**	**M1**	**M2**	**M3**
49	0,06	0,26	0,05	0,07	0,32	0,06	0,06	0,31	0,06
245	0,07	0,43	0,08	0,08	0,54	0,10	0,08	0,53	0,10
490	0,08	0,92	0,13	0,12	0,82	0,15	0,12	0,81	0,15
980	0,08	0,78	0,15	0,16	1,26	0,19	0,16	0,98	0,18
1470	0,10	0,84	0,16	0,21	1,74	0,24	0,23	1,74	0,24
2451	0,12	0,84	0,20	0,31	2,27	0,32	0,31	2,21	0,35
3922	0,13	0,85	0,25	0,43	3,15	0,46	0,18	2,28	0,26
4903	0,13	1,40	0,31	0,50	3,60	0,54	0,49	3,65	0,53

Tabla 5. Tamaño de muestra (#) vs tiempo de ejecución del requerimiento (ms).

Gráficas


Requerimientos individuales (máquina 1)
Tiempo de ejecución (ms)
0
0,1
0,2
0,3
0,4
0,5
0,6
0
1000
2000
3000
4000
5000
6000
Tamaño de muestra
Requerimiento 3 (ms)
Requerimiento 4 (ms)
Requerimiento 5 (ms)
Lineal (Requerimiento 3 (ms))
Lineal (Requerimiento 4 (ms))
Lineal (Requerimiento 5 (ms))


Requerimientos individuales (máquina 2)
Tiempo de ejecución (ms)
0
0,5
1
1,5
2
2,5
3
3,5
4
0
1000
2000
3000
4000
5000
6000
Tamaño de muestra
Requerimiento 3 (ms)
Requerimiento 4 (ms)
Requerimiento 5 (ms)
Lineal (Requerimiento 3 (ms))
Lineal (Requerimiento 4 (ms))
Lineal (Requerimiento 5 (ms))


Requerimientos individuales (máquina 3)
Tiempo de ejecución (ms)
0
0,1
0,2
0,3
0,4
0,5
0,6
0
1000
2000
3000
4000
5000
6000
Tamaño de muestra
Requerimiento 3 (ms)
Requerimiento 4 (ms)
Requerimiento 5 (ms)
Lineal (Requerimiento 3 (ms))
Lineal (Requerimiento 4 (ms))
Lineal (Requerimiento 5 (ms))

Análisis de resultados – Requerimientos individuales

En las gráficas se puede ver que el tiempo de ejecución de los requerimientos 3, 4 y 5 se comporta de forma lineal con respecto al tamaño de la muestra. Esto es lo que se esperaba inicialmente porque, como se dijo en el análisis teórico, estos requerimientos tienen una complejidad temporal de O(n). Por tanto, el valor obtenido en las pruebas coincide con el teórico.

Pruebas requerimientos avanzados (6, 7 y 8)

Descripción de las pruebas realizadas

Las pruebas realizadas para estos requerimientos son solo de complejidad temporal. Como cada uno de estos requerimientos tienen parámetros diferentes, las pruebas se realizaron de diferente manera. Las pruebas realizadas para cada requerimiento se ven así:

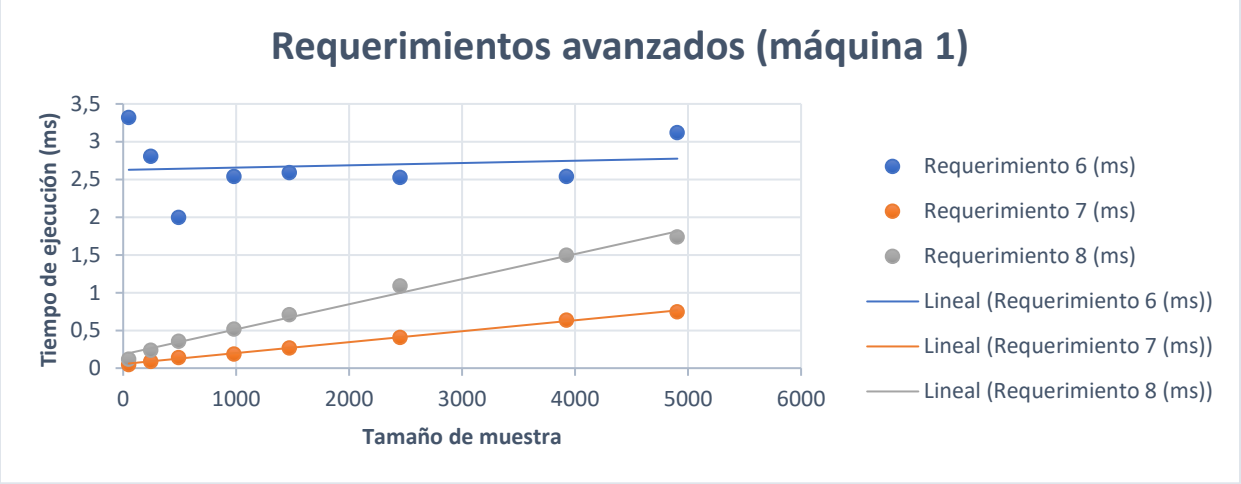
- Requerimiento 6: Como este requerimiento solo tiene el año como parámetro, se decidió fijar un año (2021) y se midió cómo cambiaba el tiempo de ejecución del requerimiento cuando se cambiaba el tamaño de la muestra.
- Requerimiento 7: Como este requerimiento tiene como parámetros año, código de subsector económico y número de actividades (top N a identificar), se decidió fijar un año (2021), un código de subsector (3) y un número de actividades a identificar (Top 10) y se midió cómo cambiaba el tiempo de ejecución del requerimiento cuando se cambiaba el tamaño de la muestra.
- Requerimiento 8: Como este requerimiento tiene como parámetros año y número de actividades (top N a identificar), se decidió fijar un año (2021) y un número de actividades (Top 10) y se midió cómo cambiaba el tiempo de ejecución del requerimiento cuando se cambiaba el tamaño de la muestra.

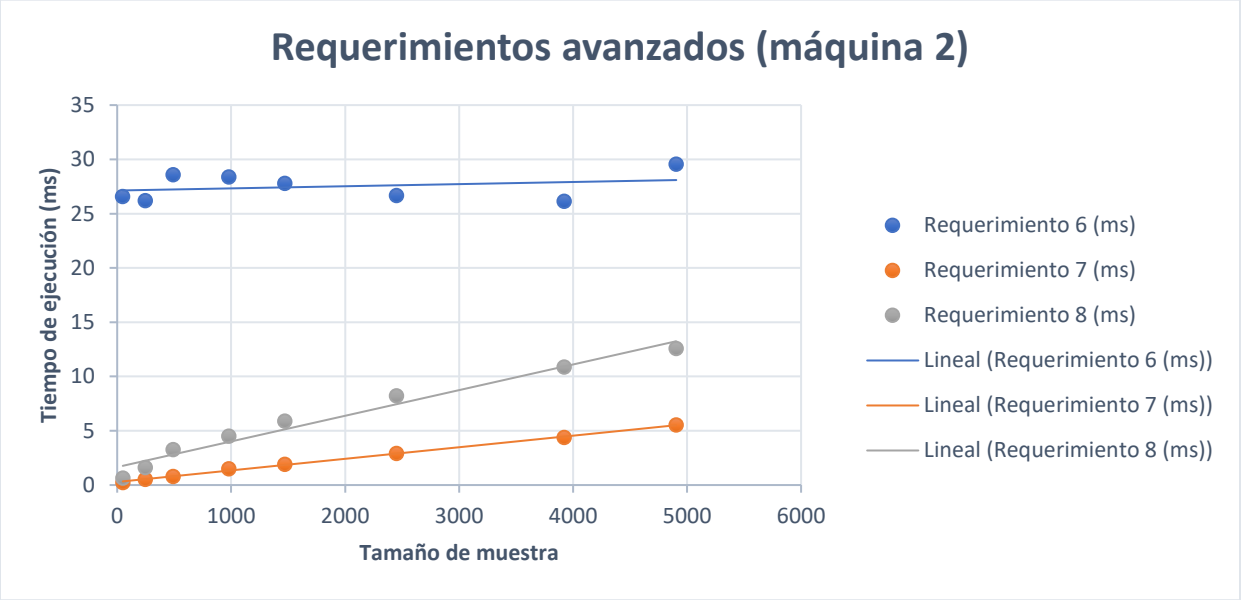
Tablas de datos

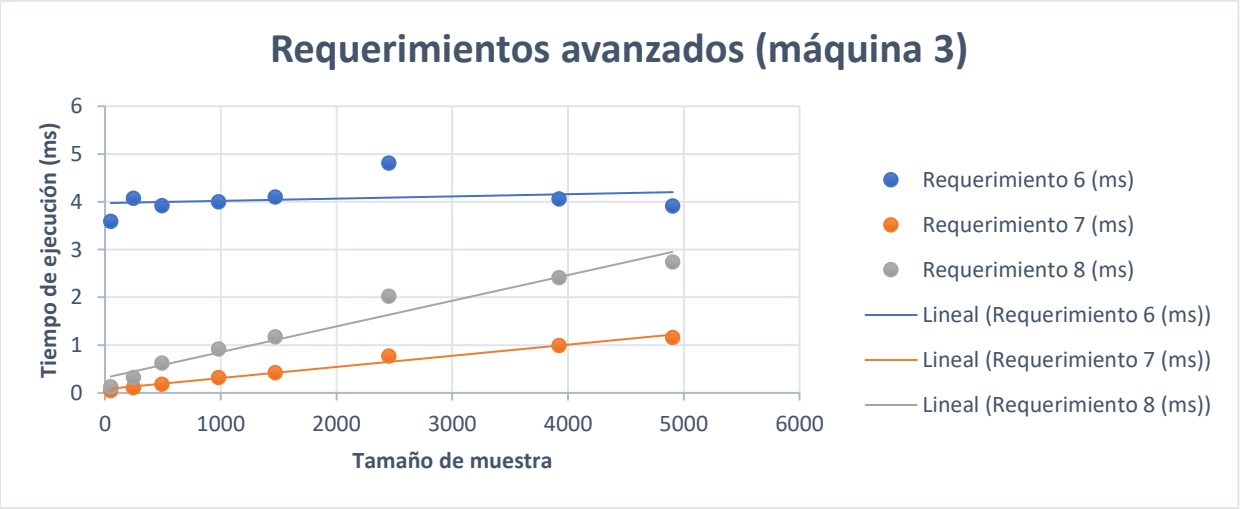
Tamaño de la muestra	**Requerimiento 6 – Tiempo (ms)**			**Requerimiento 7 – Tiempo (ms)**			**Requerimiento 8 – Tiempo (ms)**		
Maquinas	**M1**	**M2**	**M3**	**M1**	**M2**	**M3**	**M1**	**M2**	**M3**
49	3,32	26,57	3,59	0,05	0,22	0,05	0,12	0,64	0,13
245	2,81	26,21	4,07	0,09	0,51	0,11	0,24	1,60	0,32
490	2,65	28,58	3,92	0,14	0,80	0,18	0,36	3,25	0,62
980	2,54	28,38	4,00	0,19	1,48	0,32	0,52	4,50	0,92
1470	2,59	27,78	4,10	0,27	1,92	0,42	0,71	5,90	1,17
2451	2,53	26,66	4,81	0,41	2,91	0,77	1,09	8,21	2,02
3922	2,54	26,15	4,06	0,64	4,38	0,99	1,5	10,87	2,41
4903	3,12	29,57	3,91	0,75	5,53	1,16	1,74	12,57	2,74

Tabla 6. Tamaño de muestra (#) vs tiempo de ejecución del requerimiento (ms).

Gráficas


Requerimientos avanzados (máquina 1)
Tiempo de ejecución (ms)
3,5
3
2,5
2
1,5
1
0,5
0
0
1000
2000
3000
4000
5000
6000
Tamaño de muestra
Requerimiento 6 (ms)
Requerimiento 7 (ms)
Requerimiento 8 (ms)
Lineal (Requerimiento 6 (ms))
Lineal (Requerimiento 7 (ms))
Lineal (Requerimiento 8 (ms))


Requerimientos avanzados (máquina 2)
Tiempo de ejecución (ms)
35
30
25
20
15
10
5
0
0
1000
2000
3000
4000
5000
6000
Tamaño de muestra
Requerimiento 6 (ms)
Requerimiento 7 (ms)
Requerimiento 8 (ms)
Lineal (Requerimiento 6 (ms))
Lineal (Requerimiento 7 (ms))
Lineal (Requerimiento 8 (ms))


Requerimientos avanzados (máquina 3)
Tiempo de ejecución (ms)
6
5
4
3
2
1
0
0
1000
2000
3000
4000
5000
6000
Tamaño de muestra
Requerimiento 6 (ms)
Requerimiento 7 (ms)
Requerimiento 8 (ms)
Lineal (Requerimiento 6 (ms))
Lineal (Requerimiento 7 (ms))
Lineal (Requerimiento 8 (ms))

Análisis de resultados – Requerimientos avanzados

En las gráficas se puede ver que el tiempo de ejecución de los requerimientos 7 y 8 se comporta de forma lineal con respecto al tamaño de la muestra. Esto es lo que se esperaba inicialmente porque, como se dijo en el análisis teórico, estos requerimientos tienen una complejidad temporal de O(n). Por tanto, el valor obtenido en las pruebas coincide con el teórico.

En lo que respecta al requerimiento 6, se puede observar que el comportamiento es lineal, pero está muy cerca de ser constante. Esto también coincide con el análisis teórico, en el cual se afirmó que la complejidad teórica era O(n), pero en la práctica el comportamiento se debería acercar mucho a O(1).

Conclusión

Luego de este análisis exhaustivo, la revisión detallada de resultados y el trabajo en equipo para la construcción de este proyecto, podemos afirmar que este trabajo nos permitió apropiarnos profundamente de los temas vistos en este módulo del curso: las tablas de hash; así como también nos llevó a mejorar nuestro criterio y técnicas para abordar el tratamiento de los datos implementando las estructuras del módulo anterior (las listas).

Podríamos decir que la conclusión general más importante es que el entendimiento de la naturaleza de los datos y el esfuerzo por diseñar una estrategia eficiente para tratarlos es la mejor opción para llevar a cabo soluciones efectivas y sostenibles en el tiempo. A esto se le suma la voluntad y la energía de un gran equipo de trabajo, en donde todos pudimos aportar y complementar el trabajo de los demás. Terminamos con un desarrollo exitoso y fructífero para el Reto 2, preparados para todos los otros retos que vengan.