OBSERVACIONES DEL LA PRÁCTICA

Juan David Vargas Laverde, 202210212
Nicolas Ospino Zarabanda, 202211254
Nicolás Buitrago García, 202223507

	Máquina 1	Máquina 2	Máquina 3
Procesadores	Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz 1.19 GHz	Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz 2.30 GHz	M1 3.2 GHz 3,9 Ghz
Memoria RAM (GB)	12,0 GB		
Sistema Operativo	Windows 10 Sistema operativo de 64 bits	Windows 11 Home Single Language de 64 bits	MacBook Air macOS BigSur 64 bits

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

Máquina 1

Resultados

Carga de Catálogo PROBING

Factor de Carga (PROBING)	Consumo de Datos [kB]	Tiempo de Ejecución Real @LP [ms]
0.1	21984.60 kB	561.36 ms
0.5	21949.49 kB	637.72 ms
0.7	21947.60 kB	458.92 ms
0.9	21946.10 kB	439.70 ms

Tabla 2. Comparación de consumo de datos y tiempo de ejecución para carga de catálogo con el índice por categorías utilizando PROBING en la Maquina 1.

Carga de Catálogo CHAINING

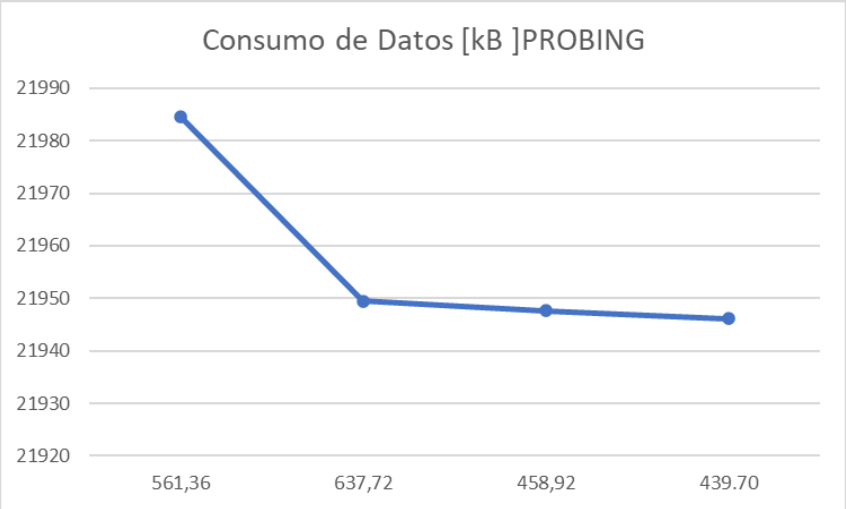
Factor de Carga (CHAINING)	Consumo de Datos [kB]	Tiempo de Ejecución Real @SC [ms]
2.00	21979.23 kB	517.11 ms
4.00	21977.57 kB	611.25 ms
6.00	21977.04 kB	615.53 ms
8.00	21976.48 kB	534.31 ms

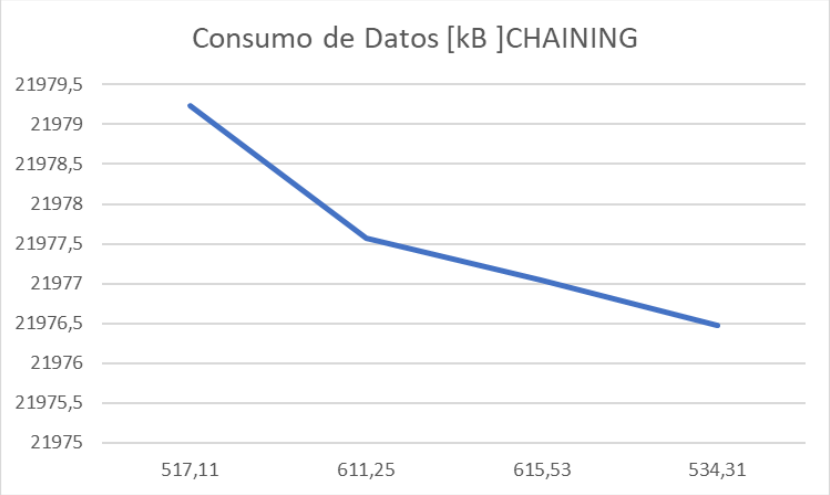
Tabla 3. Comparación de consumo de datos y tiempo de ejecución para carga de catálogo con el índice por categorías utilizando CHAINING en la Maquina 1.

Graficas

La gráfica generada por los resultados de las pruebas de rendimiento en la **Maquina 1.**

- Comparación de memoria y tiempo de ejecución para PROBING y CHAINING





Maquina 2

Resultados

Carga de Catálogo PROBING

Factor de Carga (PROBING)	Consumo de Datos [kB]	Tiempo de Ejecución Real @LP [ms]
0.1	21824.534	686.375
0.5	21949.337	457.773
0.7	21802.307	633.680
0.9	21791.690	453.885

Tabla 4. Comparación de consumo de datos y tiempo de ejecución para carga de catálogo con el índice por categorías utilizando PROBING en la Maquina 2.

Carga de Catálogo CHAINING

Factor de Carga (CHAINING)	Consumo de Datos [kB]	Tiempo de Ejecución Real @SC [ms]
2.00	21866.840	488.187

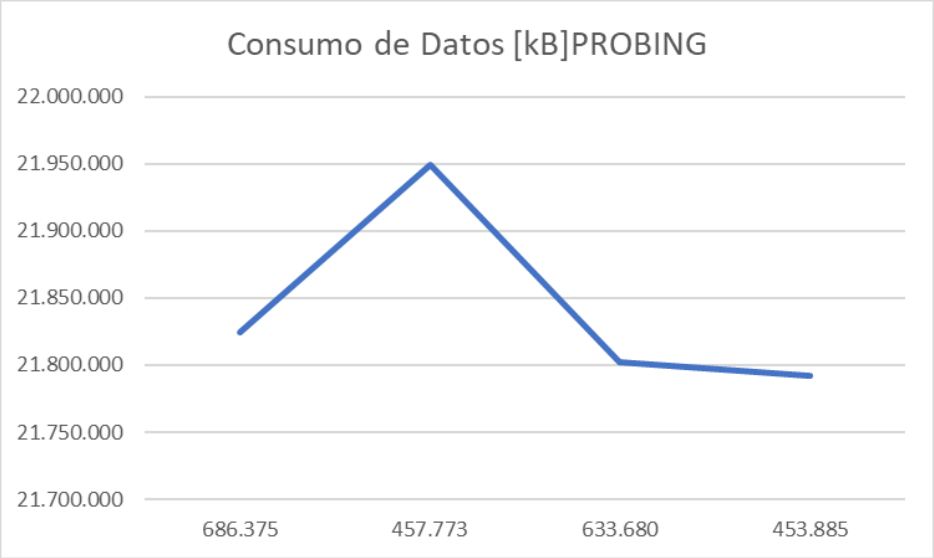
4.00	21792.781	438.723
6.00	21792.409	476.529
8.00	21791.940	436.752

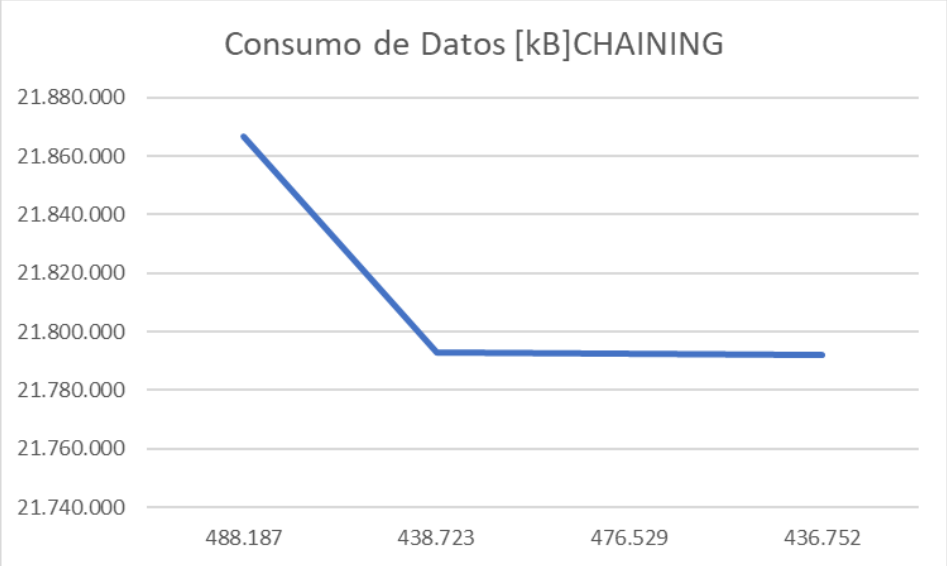
Tabla 5. Comparación de consumo de datos y tiempo de ejecución para carga de catálogo con el índice por categorías utilizando CHAINING en la Maquina 2.

Graficas

La gráfica generada por los resultados de las pruebas de rendimiento en la **Maquina 2.**

- Comparación de memoria y tiempo de ejecución para PROBING y CHAINING





Maquina 3

Resultados

Carga de Catálogo PROBING

Factor de Carga (PROBING)	Consumo de Datos [kB]	Tiempo de Ejecución Real @LP [ms]
0.1	21696.918	510.821
0.5	22011.112	439.678
0.7	21782.549	515.014
0.9	21863.207	420.301

Tabla 6. Comparación de consumo de datos y tiempo de ejecución para carga de catálogo con el índice por categorías utilizando PROBING en la Maquina 3.

Carga de Catálogo CHAINING

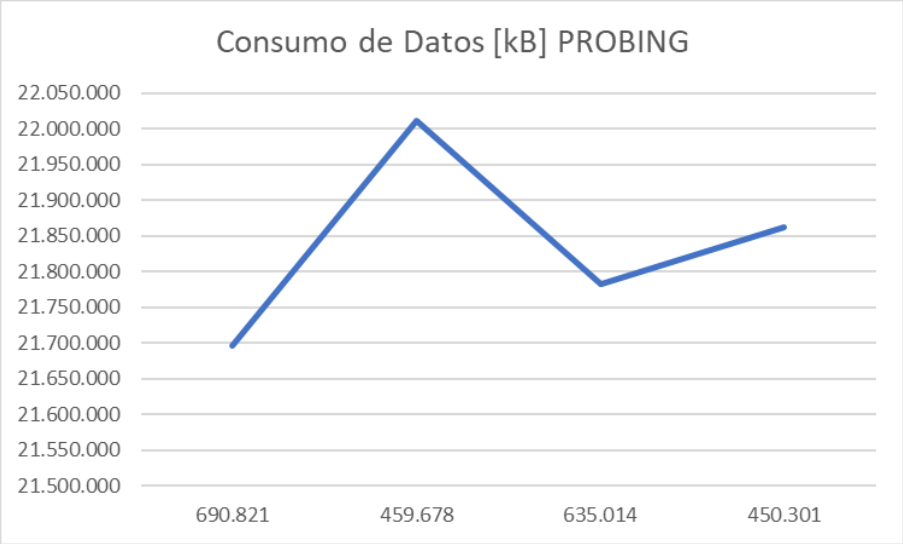
Factor de Carga (CHAINING)	Consumo de Datos [kB]	Tiempo de Ejecución Real @SC [ms]
2.00	21878.320	215.891
4.00	21785.090	363.660
6.00	21796.001	328.301
8.00	21787.629	344.116

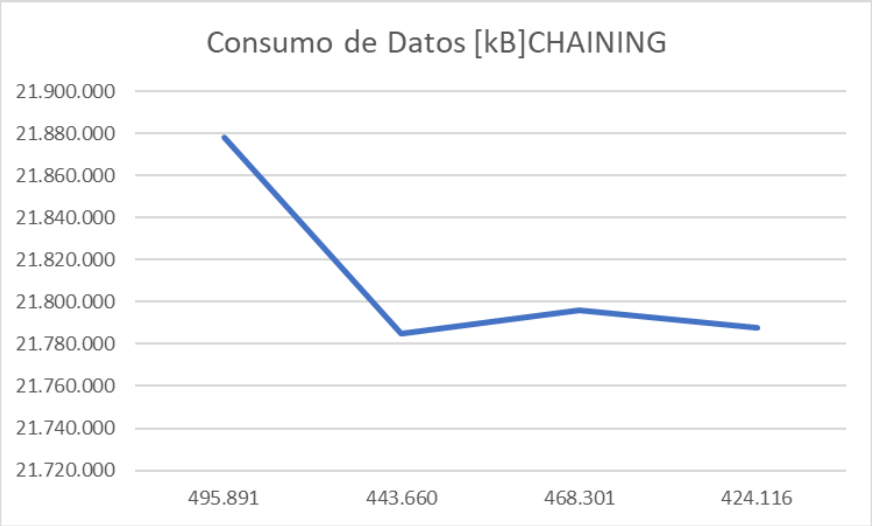
Tabla 7. Comparación de consumo de datos y tiempo de ejecución para carga de catálogo con el índice por categorías utilizando CHAINING en la Maquina 3.

Graficas

La gráfica generada por los resultados de las pruebas de rendimiento en la **Maquina 3.**

- Comparación de memoria y tiempo de ejecución para PROBING y CHAINING





Preguntas de análisis

1. ¿Por qué en la función **getTime()** se utiliza **time.perf_counter()** en vez de otras funciones como **time.process_time()**?

En la función getTime() se utiliza time.perf_counter() en vez de otras funciones como time.process_time() porque time.perf_counter() mide el tiempo de forma mas precisa y consistente. Esta función mide el tiempo en segundos con una precisión de 1 microsegundo, por tal motivo es más usada para términos de calidad.

2. ¿Por qué son importantes las funciones **start()** y **stop()** de la librería **tracemalloc**?

Estas dos funciones son importantes porque estas nos dan a conocer y nos permiten medirla cantidad de memoria que usa un programa determinado, por tal razón, ayuda a los desarrolladores a encontrar problemas relacionados con memoria. También le dan un seguimiento a la memoria para poder optimizarla.

3. ¿Por qué no se puede medir paralelamente el **uso de memoria** y el **tiempo de ejecución** de las operaciones?

El uso de la memoria y el tiempo de ejecución de la operación no se pueden medir en paralelo porque medir ambas métricas implica el uso de recursos del sistema que no se pueden compartir entre los dos tipos de medidas.

Esto requiere acceso exclusivo a los recursos de la memoria del sistema, lo que significa que cualquier otra operación que intente acceder a la memoria durante la medición puede afectar los resultados.

4. Teniendo en cuenta cada uno de los requerimientos del reto ¿Cuántos índices implementaría en el Reto? y ¿Por qué?

   Implementar dos índices el primero sería linear Probing y el otro sería Separate Chaining, en los primeros requerimientos pondremos la estructura de Probing ya que este nos permite organizar mejor la información, y en resto usaremos Chaining, esto con fines de hacer la aplicación más óptima y funcional.

5. Según los índices propuestos ¿en qué caso usaría **Linear Probing** o **Separate Chaining** en estos índices? y ¿Por qué?

   Usaría Linear Probing cuando la tabla hash es pequeña y la carga es baja. Esto se debe a que todos los elementos se almacenan en un solo arreglo y no hay necesidad de seguir punteros a través de listas enlazadas

   Usaría Separate Chaining en casos donde haya un gran número de colisiones en la tabla hash. Debido a que los elementos se insertan en una lista enlazada, no es necesario buscar en toda la tabla hash para encontrar un elemento específico, lo que lo hace más eficiente que otras técnicas de resolución de colisiones.

6. Dado el número de elementos de los archivos del reto (large), ¿Cuál sería el factor de carga para estos índices según su mecanismo de colisión?

carga para estos índices según su mecanismo de colisión?

Para determinar el factor de carga en un mecanismo de colisión, se necesita saber el tamaño de la tabla hash utilizada para almacenar los elementos. Supongamos que se utiliza una tabla hash de tamaño 5000 para almacenar los 4904 elementos de los archivos.

Factor de carga = / Factor de carga = 4904 / 5000

Por lo tanto, el factor de carga para los índices en este mecanismo de colisión sería de 0.98. Este valor se encuentra en un rango moderado de factor de carga, lo que significa que la tabla hash no está demasiado llena ni demasiado vacía.

7. ¿Qué cambios percibe en el **tiempo de ejecución** al modificar el factor de carga máximo para cargar el catálogo de contenido Streaming?

   El tiempo de ejecución es menor tanto en el probing como en el chaining cuando ponemos al máximo el factor de carga

8. ¿Qué cambios percibe en el **consumo de memoria** al modificar el factor de carga máximo para cargar el catálogo de contenido Streaming?

   El consumo de energía es menor tanto en el probing como en el chaining cuando ponemos al máximo el factor de carga

9. ¿Qué cambios percibe en el **tiempo de ejecución** al modificar el esquema de colisiones?, si los percibe, describa las diferencias y argumente su respuesta.

   En base a las pruebas que hicimo nos dimos cuenta que el tiempo de ambos es muy similar a algunos les fue mejor con el probing y a otros con el chaining

10. ¿Qué cambios percibe en el **consumo de memoria** al modificar el esquema de colisiones?, si los percibe, describa las diferencias y argumente su respuesta.

    Aquí nos dimos cuenta que el probing gasta menos memoria y como tal no encontramos otro diferencia

11. ¿Qué configuración de ideal ADT Map escogería para el **índice de años ("Año")** ?, especifique el mecanismo de colisión, el factor de carga y el número inicial de elementos.

    mecanismo de colisión probing debido a que gasta menos memoria , factor de carga el máximo debido a que es el mas rapido segun las pruebas que hicimos y número inicial de elementos debido a que los años van desde 2012 hasta 2021 9 elementos