Laboratorio 7

Hecho por:

Juan Sebastian Castro 201813107
Javier Santiago Bernal 202210546
Maria Paula Ospina 202123208

Preguntas de Observación

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

  En la función se implementa perf_counter dado que para tomar el tiempo de ejecución de una función de menor tamaño y con menos datos es efectivo utilizar una que se demore menos en completar y además sea más precisa, en este caso perf_counter cambia 500,982 en un loop que itera 1,000,000 veces y se demora 0.2427 segundos en completar. Mientras que process_time() como incluye la ejecución del programa en el sistema y ademas el tiempo del CPU durante el proceso, cambia 22 veces y se demora 0.3281 segundos en completar, además es menos precisa y es viable para funciones con gran tamaño de datos y más complejas.

- **¿Por qué son importantes las funciones start() y stop() de la librería tracemalloc?**
  La función start() de la librería tracemalloc es importante para empezar a rastrear las asignaciones de memoria en Python, asímismo la función stop() hace que se detenga el rastreo de memoria, también limpia los rastros de memoria recolectados desde el start().

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

  Dado que se deben almacenar los datos de la memoria y el tiempo en diferentes variables, y los cálculos para encontrar cada uno son distintos. Para medir la memoria se debe inicializar con start(), luego recoger la memoria en ese instante con la función previamente creada de getmemory(), y después terminar con stop(), pero antes se debe recoger la memoria al final del segmento de la medición con getmemory() también. Mientras que el proceso para medir el tiempo de ejecución es distinto, se utiliza la librería time con la creación de una función llamada getTime() al inicio del fragmento (que contiene time.perfcounter()) y se vuelve a usar gettime() al final, dichos datos se guardan en variables distintas para finalmente calcular la diferencia de tiempos

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

  Implementaría un catálogo con tres llaves: Una de ellas sería “Genres” que contendría un mapa, que tendría los índices de los nombres de los géneros, sin

repetirlos. Dado que el requerimiento para la carga de datos es ordenar el catálogo por géneros que dentro contengan los títulos de ese género

Otra llave dentro del catálogo sería “Streaming” que también contendría un mapa, cuyos índices serían “Hulu”, “Disney”,”Netflix” y “Amazon” dado que otro requerimiento de la carga de datos es mostrar los primeros tres y últimos tres títulos cargados con su información de cada plataforma. Si no se crea otro mapa es difícil cumplir este requerimiento con el mapa de “Genres”, es por esto que se debe hacer otro.

La última llave sería una arraylist llamada “All” para almacenar todos los títulos de series y películas con su información, sin filtrar por géneros ni streaming.

En general, el número de índices que utilizaría para el mapa de “Genres” depende de cuántos haya en la base de datos. Y los índices que utilizaría para el mapa “Streaming” serían 4, uno para cada plataforma.

- **Según los índices propuestos ¿en qué caso usaría Linear Probing o Separate Chaining en estos índices? y ¿Por qué?** Para todos los índices usaría separate chaining dado que se tiene que categorizar la información y eso mejor tener una sóla llave con varios valores en común en ella, que tener varias llaves iguales con diferentes valores dentro. Se ordena mejor la información. Además, para el caso de los requerimientos, se requiere usar las llaves de los diferentes genres dentro del mapa, y es más fácil acceder y buscar los valores dentro con Separate Chaining

- **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?** El factor de carga de estos índices teniendo en cuenta que es separate chaining sería de 4.

Para las respuestas a las siguientes preguntas se utilizaron las tablas #2, tabla #3 y gráficas #1 y #2

- **¿Qué cambios percibe en el tiempo de ejecución al modificar el factor de carga máximo para cargar el catálogo de contenido Streaming?**
  En linear probing, cuando se cambia el factor de carga en el 0.1, 0.5 y 0.7 se va reduciendo el tiempo de ejecución desde los 1631 ms hasta los 1395 ms. Sin embargo, en el factor 0.9 la tendencia no sigue, esto se puede deber a que la recolección del tiempo en este factor de carga no estuvo bien recolectado, aunque se hicieron 3 pruebas de ejecución para cada factor de carga.

  Por otro lado, en Chaining no se ve una tendencia clara cuando se cambia el factor de carga en el tiempo de ejecución. Sin embargo, se puede evidenciar que cuando se tiene un factor de carga de 2, el tiempo de ejecución es es menor al del factor de carga de 8.00 con 1579 ms y 1600 ms respectivamente. El cambio en el factor de carga no afecta en gran medida los tiempos de ejecución, puesto que la diferencia entre el 2 y el 8 es de 21 ms.

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

  En linear probing, cuando se aumenta el factor de carga se reduce el consumo de memoria, en 0.1 el consumo de datos es 36090 kB que es mayor al factor más grande que es 0.9 con un consumo de 35636 kB. Sin embargo, la diferencia entre ambos no es muy grande, siendo 454kB.

  En Chaining, cuando se aumenta el factor de carga el consumo de datos no presenta una tendencia y se mantiene más o menos estable, siendo en promedio 35654,36 kB. El consumo de datos más grande se presenta en el factor de carga 2, mientras que el más pequeño es en el factor 8.

- **¿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 promedio, el Probing obtuvo mejor tiempo de ejecución con 1519,96ms mientras que el chaining obtuvo 1581,944 ms. Además, en el channing se refleja una tendencia clara de incremento en el tiempo de ejecución cuando se va aumentando el factor de carga, mientras que en el probing la tendencia no se ve claramente. Aunque sí se puede decir que este último decrece en tiempos de ejecución los primeros tres factores de carga, del 0.1 al 0.7
- **¿Qué cambios percibe en el consumo de memoria al modificar el esquema de colisiones?, si los percibe, describa las diferencias y argumente su respuesta.**
  En el consumo de memoria, el Chaining obtuvo un menor valor puesto que en promedio consumió 35654,36 kB, mientras que el Probing consumió en promedio 35766,05 kB. Sin embargo la diferencia entre ambos esquemas de colisiones es muy pequeña, siendo 112 kB. Se puede decir que ambos esquemas se mantienen en el mismo promedio de consumo de memoria cuando se cambian los factores de carga que es más o menos 35610,21 kB.
- **¿Qué configuración de ideal ADT Map escogería para el índice géneros de contenido (“listed_in”)?, especifique el mecanismo de colisión, el factor de carga y el numero inicial de elementos.**
  Para el índice de contenidos escogería el chaining puesto que consume menos memoria y además su tiempo de ejecución en la carga de datos no estuvo muy alto, se mantuvo casi igual que probing con 62ms de diferencia. El factor de carga sería de 0.5 puesto que tiene una relación óptima entre el tiempo de ejecución y el consumo de datos durante la carga. El número inicial de elementos es la cantidad de géneros diferentes existen en el archivo large, en este caso, el large tiene 53 genres por lo tanto ese es el número inicial de elementos.

Tabla #1: Máquinas de cómputo

	Máquina 1	***Máquina 2***	***Máquina 3***
Procesadores	Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz 2.11 GHz	Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz 2.59 GHz	

Memoria RAM (GB)	8.00 GB (7.79 GB usable)	8.00 GB	
Sistema Operativo	Windows 11 Home	Windows 11 64-bit operating system	

Máquina 1: María Paula Ospina 202123208

Tablas de uso de memoria y tiempos de ejecución

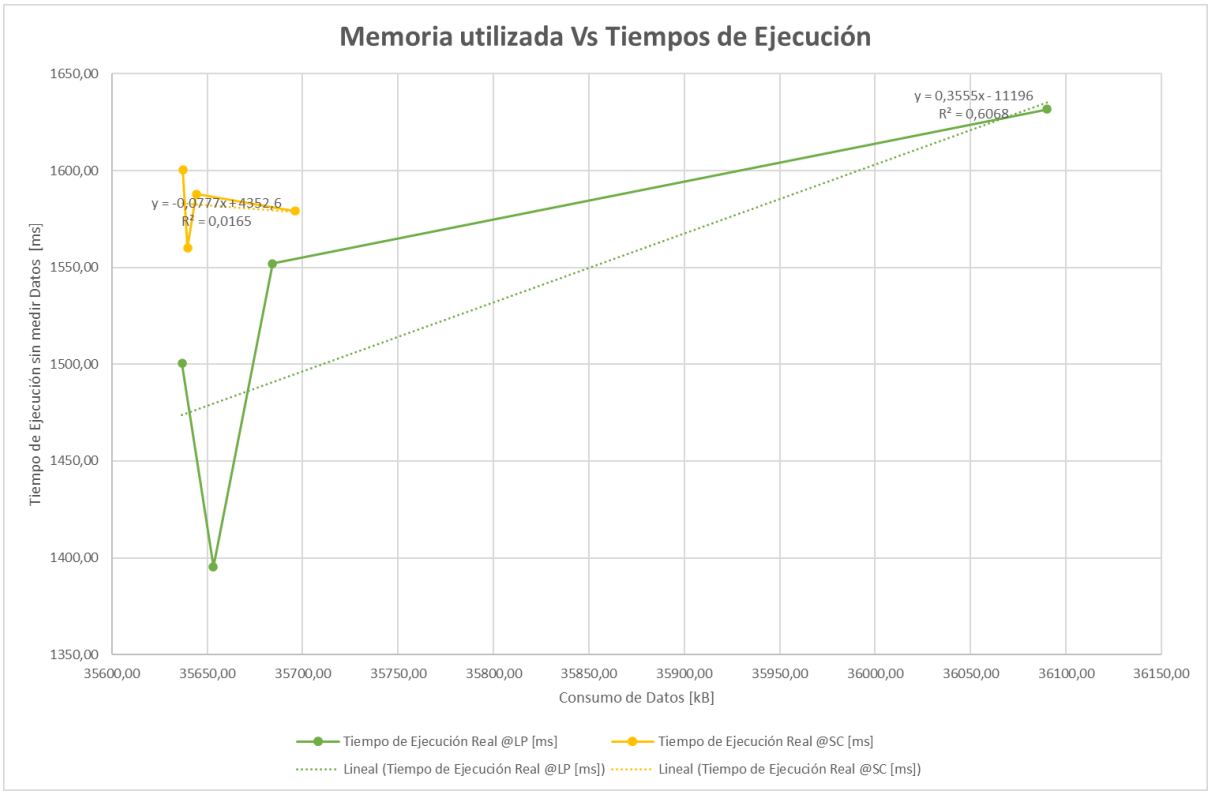
Tabla #2: Carga de Catálogo PROBING (-large)

Factor de Carga (PROBING)	Consumo de Datos [kB]	Tiempo de Ejecución Real @LP [ms]
0,1	36090,02	1631,78
0,5	35684,27	1552,01
0,7	35653,13	1395,54
0,9	35636,80	1500,53

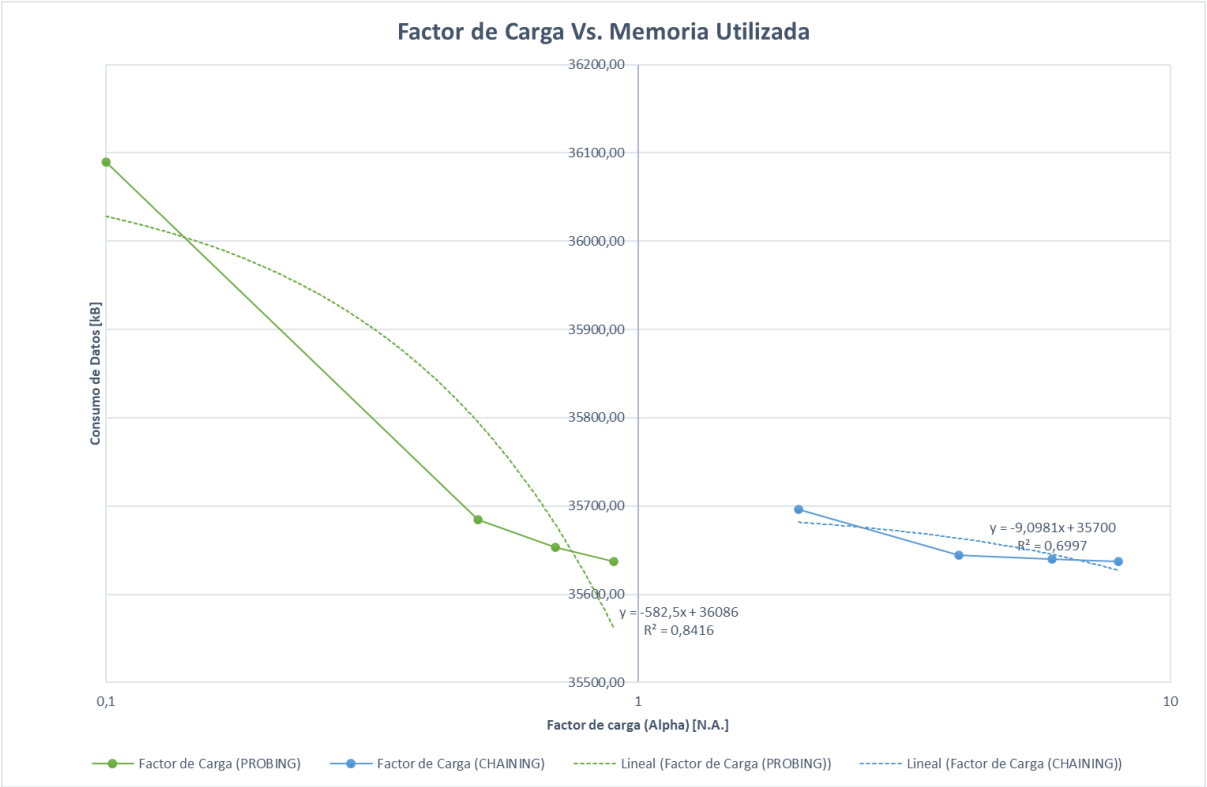
Tabla #3: Carga de Catálogo CHAINING (-large)

Factor de Carga (CHAINING)	Consumo de Datos [kB]	Tiempo de Ejecución Real @SC [ms]
2,00	35696,24	1579,05
4,00	35644,34	1587,93
6,00	35639,71	1560,21
8,00	35637,13	1600,58

Gráfica #1: Memoria utilizada vs Tiempos de Ejecución



Gráfica #2: Factor de Carga vs Memoria Utilizada



Máquina 2: Javier Santiago Bernal 202210546

Tablas de uso de memoria y tiempos de ejecución

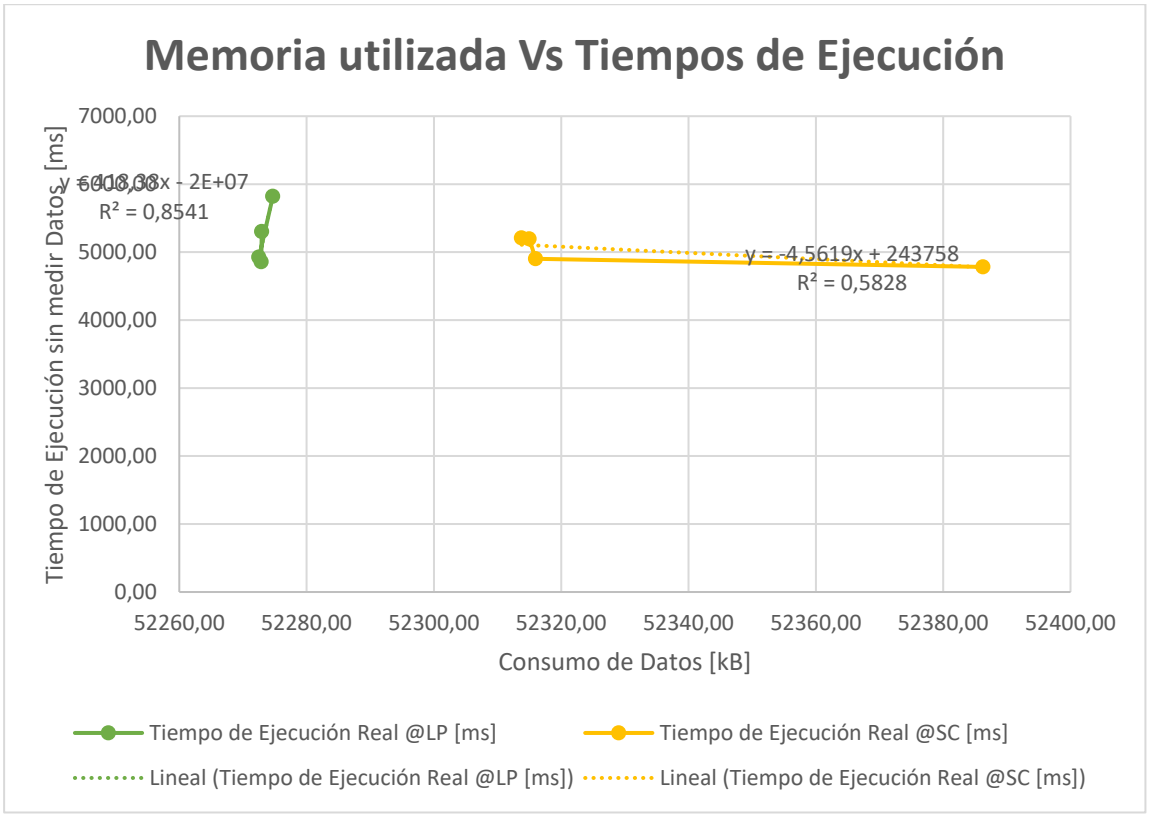
Tabla #2: Carga de Catálogo PROBING (-large)

Factor de Carga (PROBING)	**Consumo de Datos [kB]**	**Tiempo de Ejecución Real @LP [ms]**
0,1	52274,70	5821,12
0,5	52272,52	4925,62
0,7	52272,88	4858,65
0,9	52272,96	5304,18

Tabla #3: Carga de Catálogo CHAINING (-large)

Factor de Carga (CHAINING)	**Consumo de Datos [kB]**	**Tiempo de Ejecución Real @SC [ms]**
2,00	52386,28	4782,32
4,00	52315,98	4902,36
6,00	52314,98	5195,53
8,00	52313,79	5208,01

Gráfica #1: Memoria utilizada vs Tiempos de Ejecución



Gráfica #2: Factor de Carga vs Memoria Utilizada