OBSERVACIONES DEL LA PRACTICA

Estudiante 1: Santiago Castro Arciniegas Cod 202014994
Estudiante 2: Maria Camila Luna Velasco Cod 201920993

	Máquina 1	**Máquina 2**
Procesadores	AMD Ryzen 7 3700U with Radeon Vega Mobile Gfx 2.30 GHz	AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx 2.10 GHz
Memoria RAM (GB)	8 GB	8 GB
Sistema Operativo	Windows 10 Home 64 bits	Windows 10 Home 64 bits

Maquina 1

Resultados

Carga de Catálogo PROBING

Factor de Carga (PROBING)	**Consumo de Datos [kB]**	**Tiempo de Ejecución [ms]**
0,30	1307650,355	54539,884
0,50	1307650,574	56033,996
0,80	1307650,355	57098,547

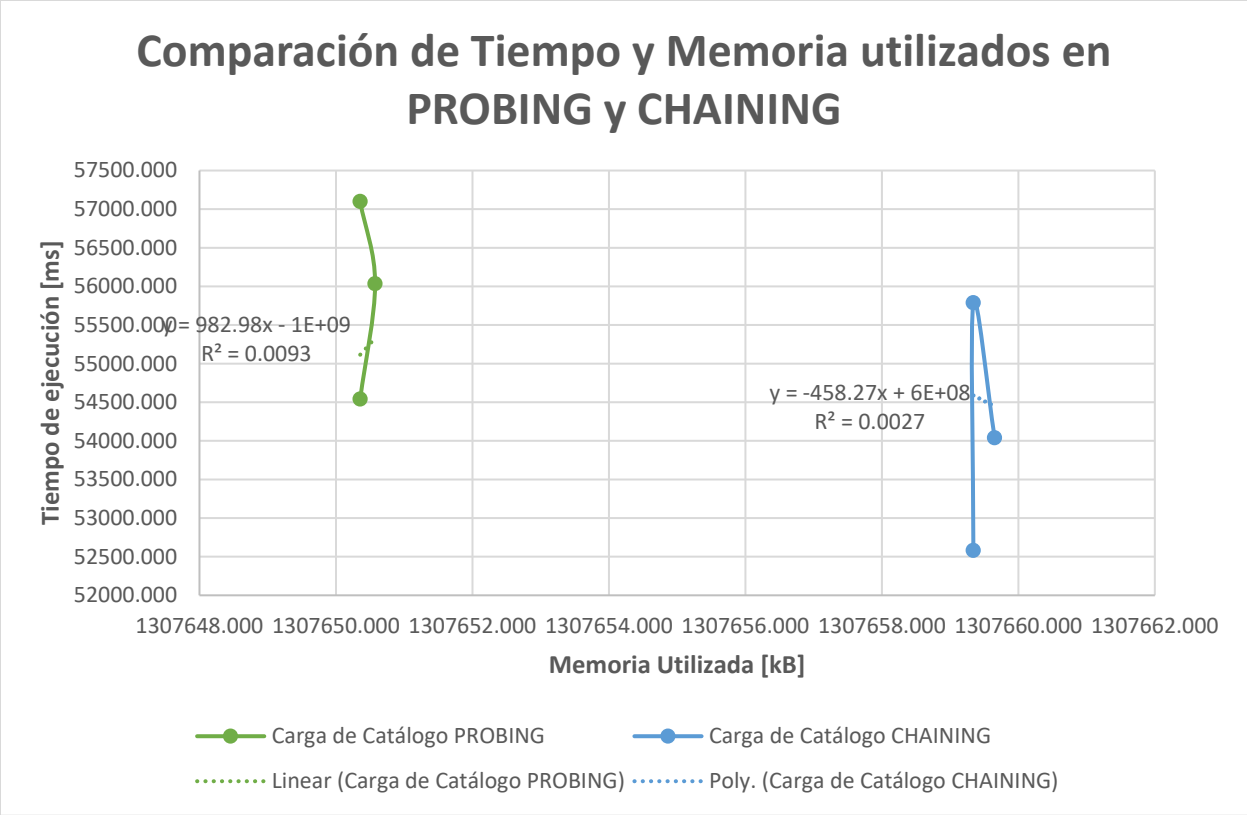
Carga de Catálogo CHAINING

Factor de Carga (CHAINING)	**Consumo de Datos [kB]**	**Tiempo de Ejecución [ms]**
2,00	1307659,653	54038,963
4,00	1307659,341	55787,808
6,00	1307659,341	52579,867

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 [ms]**
0.30	1307650,297	51510,395
0.50	1307650,574	54145,263
0.80	1307650,355	48312,456

Tabla 1. 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

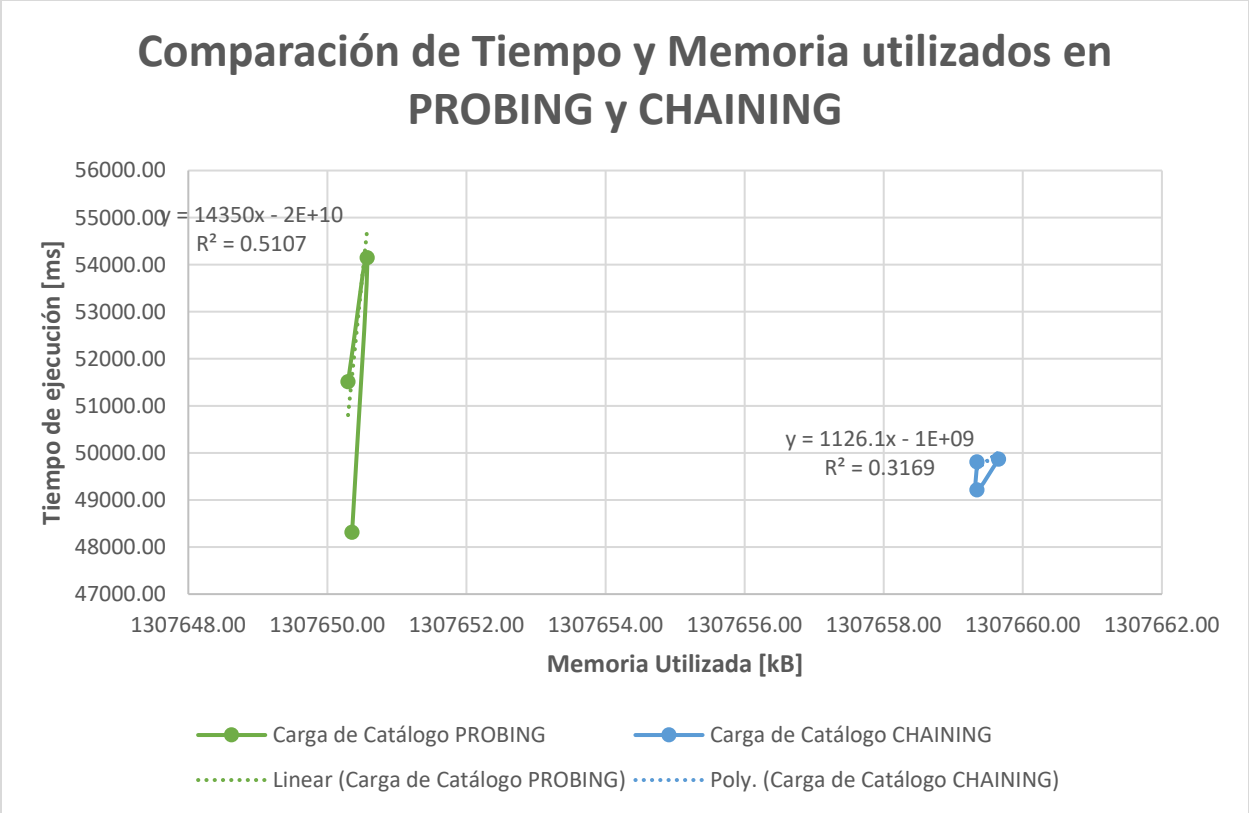
Factor de Carga (CHAINING)	**Consumo de Datos [kB]**	**Tiempo de Ejecución [ms]**
2.00	1307659,653	49865,765
4.00	1307659,341	49210,874
6.00	1307659,341	49808,665

Tabla 2. 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



Preguntas de análisis

1) ¿Por qué en la función **getTime()** se utiliza **time.perf_counter()** en ves de la previamente conocida **time.process_time()**?

   perf_counter mide el tiempo real que toma llevar a cabo un proceso; mientras que process_time retorna el tiempo total gastado por el computador para el proceso, es decir, no tiene en cuenta el tiempo que el ordenador se gasta haciendo cualquier otra cosa. perf_counter suele ser preferible pero process_time puede ser útil si se quiere comparar la eficiencia de código.

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

   El start y el stop permiten, respectivamente, iniciar y terminar de rastrear la memoria que gasta Python en el computador.

3) ¿Qué cambios percibe en el **tiempo de ejecución** al modificar el factor de carga máximo para cargar el catálogo de videos?

   Al parecer, disminuyó el tiempo consumido para Chaining en la máquina 1, pero aumentó para Probing; mientras que para la máquina 2 disminuyó para Probing, pero no varió mucho para chaining.

4) ¿Qué cambios percibe en el **consumo de memoria** al modificar el factor de carga máximo para cargar el catálogo de videos?

   No hay una diferencia significativa al cambiar el factor de carga en ninguna de las dos máquinas.

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

   No percibimos cambios significativos en el tiempo de ejecución al cambiar de esquema de colisiones.

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

   Se notó una diferencia significativa en ambas máquinas al cambiar de Probing (menor consumo) a Chaining (mayor consumo).

   Aclaración: Los datos medidos son poco concluyentes porque la muestra no es significativa, solo se obtuvo un valor por factor de carga y esquema de colisiones; además, solo se midió con 3 factores distintos por esquema.