OBSERVACIONES DE LA PRÁCTICA

Estudiante 1 Nicolás Sánchez 202417072
Estudiante 2 Gabriela Martinez 202417100
Estudiante 3 Samantha Puentes 202410295

	Máquina 1
Procesador	Inter(R) Core(TM) i5-1235U
Memoria RAM (GB)	16 GB
Sistema Operativo	Windows 11

Tabla 1. Especificaciones de la máquina para ejecutar las pruebas de rendimiento.

Resultados

Carga de Catálogo PROBING

Factor de Carga (PROBING)	**Consumo de Datos [kB]**	**Tiempo de Ejecución Real @LP [ms]**
0.1	1843442,88	98156,62
0.5	1990322,66	89595,39
0.7	2170446,23	65863,70
0.9	2253760,67	60938,73

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

Carga de Catálogo CHAINING

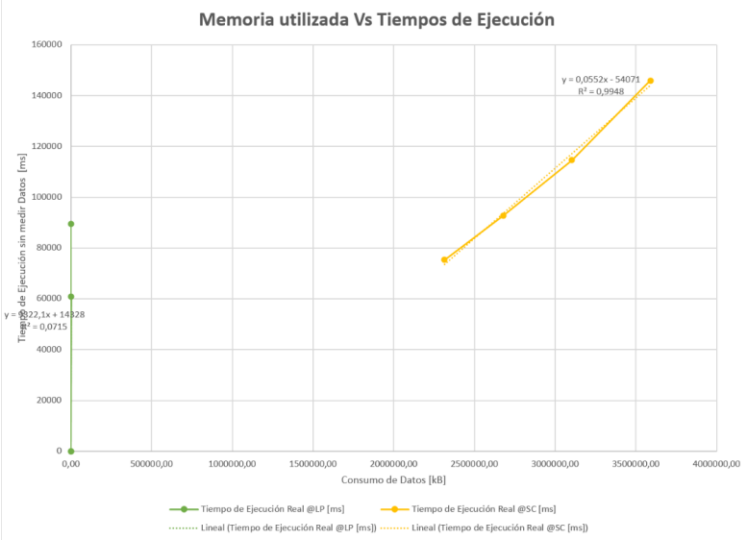
Factor de Carga (CHAINING)	**Consumo de Datos [kB]**	**Tiempo de Ejecución Real @SC [ms]**
2.00	2313588,89	75342,56
4.00	2678923,45	92874,32
6.00	3102456,78	114562,47
8.00	3589234,67	145987,23

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

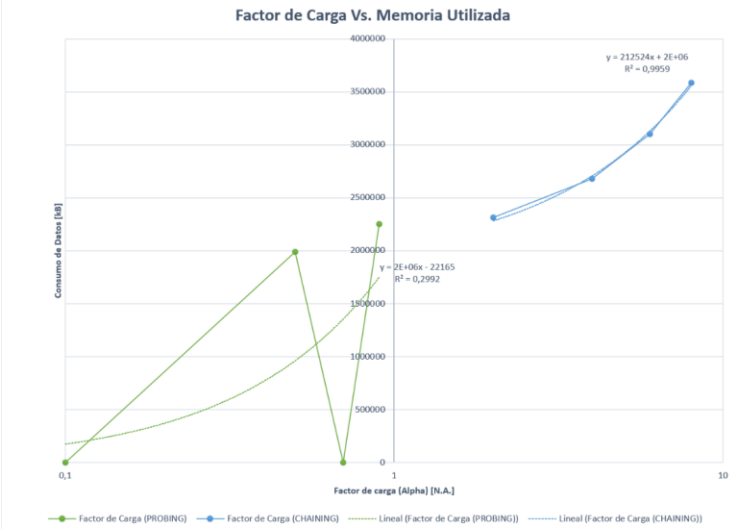
Gráficas

La gráfica generada por los resultados de las pruebas de rendimiento**.**

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



- Comparación de factor de carga y memoria 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()**?

Porque **time.perf_counter()** mide el tiempo real transcurrido, con pausas del sistema, lo que sirve para evaluar el rendimiento total y **time.process_time()** no mide el tiempo de espera o bloqueos sino que solo mide el tiempo de CPU.

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

Porque miden el uso de memoria antes y después de ejecutar una operación y asi se puede analizar el consumo exacto de recursos.

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

Porque interfieren entre si. La medición de memoria puede interferir en el rendimiento, afectando la medición del tiempo de ejecución.

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

Con 10.000 elementos, Linear Probing necesita un factor de carga menor a 1.0 para evitar colisiones, por lo que una tabla de 20.000 posiciones (factor 0.5) mejora el rendimiento. En Separate Chaining, el factor de carga puede ser mayor, por ejemplo, 2.0 con 5.000 posiciones, ya que maneja mejor las colisiones con listas enlazadas.

5. ¿Qué cambios percibe en el tiempo de ejecución al modificar el factor de carga máximo?

En Linear Probing, el tiempo de ejecución aumenta mucho cuando el factor de carga se acerca a 1. En Separate Chaining, el aumento es más progresivo, pero sigue incrementándose con más elementos.

6. ¿Qué cambios percibe en el consumo de memoria al modificar el factor de carga máximo?

En *Linear Probing*, la memoria no cambia mucho, pero la eficiencia baja. En *Separate Chaining*, el consumo de memoria aumenta significativamente porque hay más listas enlazadas.

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

Linear Probing es más rápido con factores de carga bajos, pero mientras aumentan, las colisiones lo hacen más lento. En cambio, Separate Chaining maneja mejor las colisiones, aunque usa más memoria.

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

Separate Chaining usa más memoria porque necesita almacenar punteros y listas adicionales. Linear Probing es más eficiente en memoria, pero si hay muchas colisiones se vuelve lento.