OBSERVACIONES DE LA PRÁCTICA

Santiago Rojas Cod 202420928
Luis Alejandro Rodriguez Cod 202321287

	Máquina 1
Procesador	AMD Ryzen 5 4600 H
Memoria RAM (GB)	8
Sistema Operativo	Windows 10

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

Resultados

Carga de Catálogo LINEAR PROBING

Factor de Carga (PROBING)	**Consumo de Datos [kB]**	**Tiempo de Ejecución Real @LP [ms]**
0.1		
0.5	1948845.73	677574.54
0.7	1942633.38	509106.66
0.9		

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

Carga de Catálogo SEPARATE CHAINING

Factor de Carga (CHAINING)	**Consumo de Datos [kB]**	**Tiempo de Ejecución Real @SC [ms]**
2.00	1962150.85	129771.91
4.00	1962150.13	193370.99
6.00		
8.00		

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

Gráficas

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

- Comparación de memoria y tiempo de ejecución para LINEAR PROBING y SEPARATE CHAINING
- Comparación de factor de carga y memoria para LINEAR PROBING y SEPARATE 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 perf_counter() mide todo el tiempo que se demora el programa en correr y en ejecutarse, inculyendo todo lo que pase, como lectura de archivos o pausas. Eso es lo que se quiere medir en este laboratorio en cambio time.proccess_time() mide todo lo que no incluya CPU, por lo cual va dar un tiempo menor.

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

Porque start empieza el seguimiento en el programa y a través de la ejecución, Sin llamarla no se guarda la información de la memoria en uso. Ademas por eso podemos llamar a getMemory() que es donde elegimos cuando recibir el usaje de memoria hasta ese punto del código, que eso ya es decidido por el usuario. Por lo cual los dos sirven para medir la memoria en ciertos puntos del código que uno escoja

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

Porque los dos se miden de forma independiente y se miden en diferentes momentos. SI se intentara medir al mismo tiempo se podría mezclar y no se sabría que parte del tiempo o memoria se mide en que parte o cual no se midió.

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

Basandome en lo aprendido en clase el mejor factor de carga para linear probing es menor o igual a 0.7 ya que evita que haya muchas colisiones o búsquedas grandes

En separate chaining el mejor factor de carga esta entre 2.0 y 4.0, y es más alto ya que la cantidad de elementos no afecta mucho el rendimiento.

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

Para mi ejecución todos tuvieron un tiempo de ejecución muy alta, más de 10 minutos mínimo, por lo cual me base en la lógica y solo espere por toda la carga de datos en los factores de carga recomendados ya que en los otros tomo mucho tiempo llegando a más de los 20 minutos. Pero ya cuando recibía los datos podía ver que el cambio entre factores de carga no es tan grande, y en mis pruebas de Linear Probing,

bajo de tiempo de ejecución entre mayor factor de carga. En cambio, en Separate chaining entre mayor factor de carga más tiempo de ejecución le tomaba a la carga de catálogos.

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

Con mis ejecuciones con estos factores de carga puedo ver que el cambio en memoria entre factores de carga es mínimo y ni se nota, se puede ver que aumenta entre menos factor de carga pero el cambio es mínimo, en Separate chaining sobre todo la diferencia en memoria usada es de decimales

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.

Ya que las pruebas, que hacemos en cada una son de diferentes factores de carga, no hay verdadera forma de compararlos entre sí, pero podemos asumir por lo aprendido, que Separate chaining es mejor con factores de carga altos; y con estos factores de carga recomendados separate chaining tiene tiempo de ejecución más rápido que Linear Probing con sus factores de carga recomendados. Como estas tablas son más llenas Separate Chaining es mejor que Linear Probing en este caso porque su tiempo siempre se mantiene estable, y linear probing si disminuye

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.

En el uso de memoria se puede ver como Linear Probing usa menos memoria que separate chaining cada uno respectivamente en sus factores de carga diferentes. Este es trivial ya que en Linear Probing solo se una lista principal, en cambio en separate chaining tiene un bucket por cada índice, lo que significa una lista adicional y memoria extra por cada índice por lo cual va ser mayor.