OBSERVACIONES DE LA PRÁCTICA

Estudiante 1 Cod XXXX
Estudiante 2 Cod XXXX
Estudiante 3 Cod XXXX

	Máquina 1
Procesador	Intel(R) Core(TM) i5-10310U CPU @ 1.70GHz (2.21 GHz)
Memoria RAM (GB)	16.0 GB
Sistema Operativo	64-bit operating system, x64-based processor

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	1603376.82	429045.46
0.5	562889.04	238717.11
0.7	407488.65	179389.40
0.9	323018.93	177862.75

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	1640907.53	59336.91
4.00	1636332.35	51960.45
6.00	1634838.80	47492.17
8.00	1634131.83	43214.19

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()**?

   R//: Perf_counter() mide el tiempo real transcurrido en cambio otras funciones como process_time() que mide el tiempo que toma el CPU en realizar la función sin tener en cuenta otros parámetros que afectan el tiempo real transcurrido.

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

   R//: Las funciones start() y stop() de la librería tracemalloc son importantes para activar y desactivar el rastreo y delimitar el intervalo del que se quiere medir la memoria

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

   R//: Al medir el uso de memoria se altera el tiempo medido, por eso no es correcto medir paralelamente estas dos operaciones.

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

   R//: Para Linear Probing, se recomienda un factor de carga cercano a 0.7, ya que valores mayores generan muchas colisiones y menores desperdician memoria. En cambio, para Separate Chaining, se puede usar un factor de carga más alto, entre 4 y 8, porque las colisiones se manejan en listas y no afectan tanto el rendimiento.

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

   R//: Al aumentar el factor de carga en linear probing, se puede evidenciar que el tiempo de ejecución disminuye, cuando se reduce el factor de carga. Y en separate, al aumentar el factor de carga máximo, el tiempo de ejecución disminuye. Esto ocurre porque una mayor cantidad de elementos por cubeta reduce el tamaño de la tabla y el tiempo necesario para procesarla, sin que las colisiones afecten tanto el rendimiento como en linear probing.

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

   R//: Al aumentar el factor de carga máximo, el consumo de memoria disminuye, ya que la tabla utiliza menos espacio para almacenar la misma cantidad de datos. En cambio, con un factor de carga más bajo, la tabla se vuelve más grande y requiere más memoria para mantener menos elementos y reducir colisiones. Y en separate chaining, El consumo de memoria se mantiene casi constante, con una ligera disminución al aumentar el factor de carga. Esto se debe a que al usar una tabla más compacta se aprovecha mejor la memoria.

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.

   R//: Al modificar el esquema de colisiones se observa que *Separate Chaining* tiene un menor tiempo de ejecución que *Linear Probing*. Esto ocurre porque en chaining las colisiones se manejan en listas, lo que evita los recorridos largos dentro de la tabla. En cambio, en linear probing, las posiciones consecutivas ocupadas generan más tiempo de búsqueda e inserción.

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.

   R//: El consumo de memoria, *Linear Probing* usa menos memoria cuando el factor de carga aumenta, ya que depende directamente del tamaño del arreglo. Por su parte, *Separate Chaining* mantiene un consumo más constante, porque además de la tabla necesita espacio adicional para las listas que almacenan las colisiones.