OBSERVACIONES DEL LA PRACTICA

Ana Sofía Castellanos 202114167
Martín Santiago Galván Castro 201911013

	Máquina 1	Máquina 2
Procesadores	Intel(R) Core (TM) i7-10750H CPU @ 2.60GHz 2.59 GHz	AMD Ryzen 3 3200G with Radeon Vega Graphics 3.60 GHz
Memoria RAM (GB)	16.0 GB	16.0 GB
Sistema Operativo	Windows 10 Home Single Language	Windows 10 Pro

Tabla 1. Especificaciones de las máquinas para ejecutar las pruebas de rendimiento.

Maquina 1

Resultados

Carga de Catálogo PROBING

Factor de Carga (PROBING)	Consumo de Datos [kB]	Tiempo de Ejecución [ms]
0.30	1324227,79	20249,573
0.50	1324227,79	19836,238
0.80	1324227,79	19202,86

Tabla 2. Comparación de consumo de datos y tiempo de ejecución para carga de catálogo con el índice por categorías utilizando PROBING en la Maquina 1.

Carga de Catálogo CHAINING

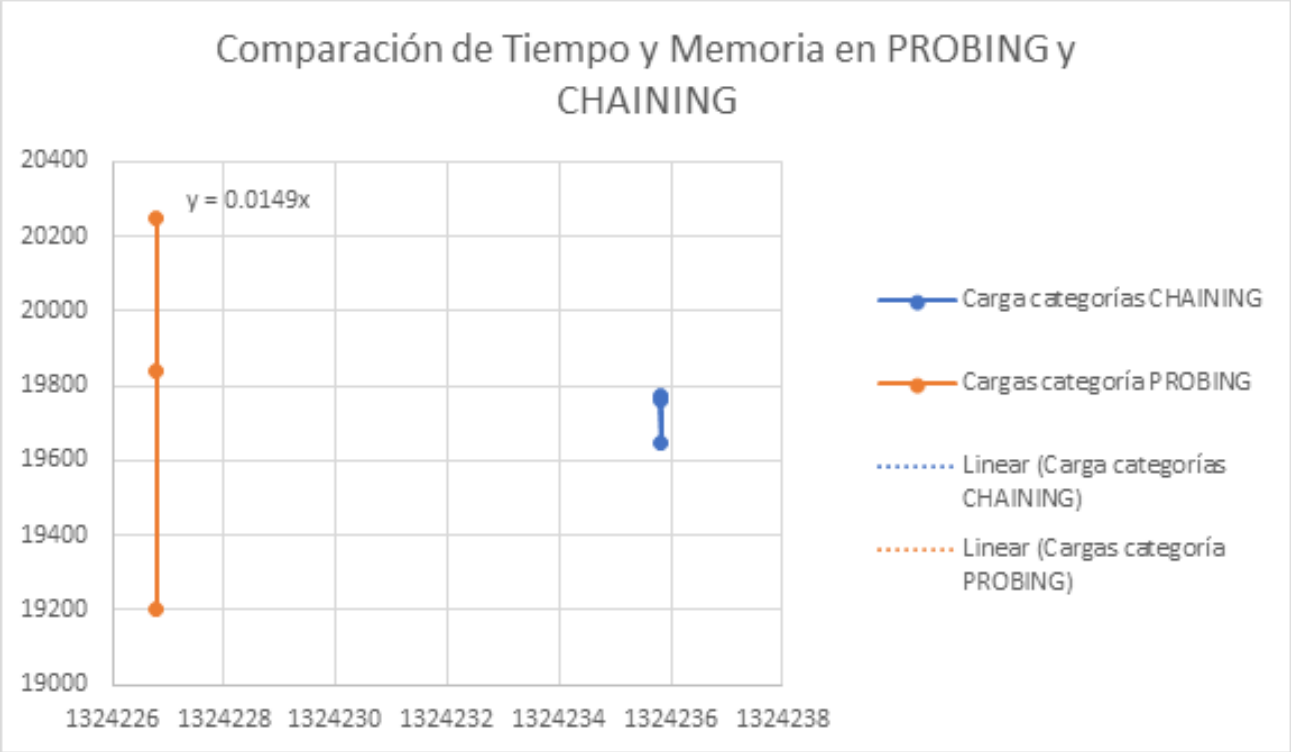
Factor de Carga (CHAINING)	Consumo de Datos [kB]	Tiempo de Ejecución [ms]
2.00	1324236,8	19758,95
4.00	1324236,8	19646,22
6.00	1324236,8	19772,18

Tabla 3. Comparación de consumo de datos y tiempo de ejecución para carga de catálogo con el índice por categorías utilizando CHAINING en la Maquina 1.

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	1307634,82	26872,76
0.50	1307634,82	26517,06
0.80	1307634,82	26082,49

Tabla 4. 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

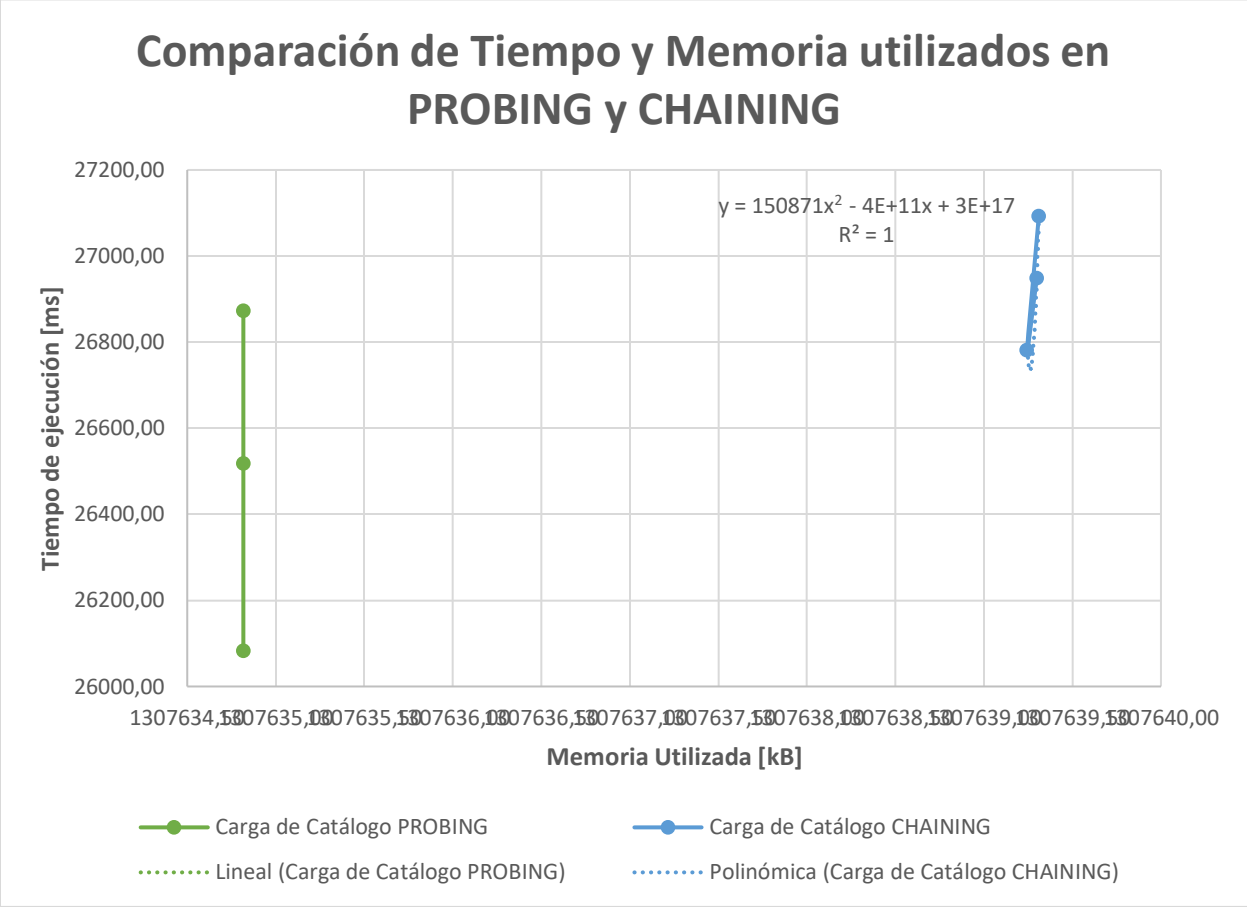
Factor de Carga (CHAINING)	**Consumo de Datos [kB]**	**Tiempo de Ejecución [ms]**
2.00	1307639,31	27091,68
4.00	1307639,24	26781,34
6.00	1307639,30	26948,68

Tabla 5. 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()**

La función perf_counter() cuenta el tiempo de manera relativa, un tiempo que no tiene relación con el mundo real, sino que surge de la relación que exista entre el sistema y el código que se está implementando. Este tiempo del perf_counter() se mide utilizando un contador de CPU el cual se incrementa a una frecuencia relacionada con el reloj de hardware de la CPU.

La función process_time() por otro lado, calcula considerando este tiempo derivado del contador de la CPU, sin embargo, se actualiza cuando el código se está ejecutando en la CPU, lo cual puede llevar a que no solo se cuente el tiempo del proceso en CPU sino también un tiempo del sistema el cual es el tiempo en el que se ejecuta parte del sistema operativo con el nombre de dicho proceso.

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

La función start() de la libraria tracemalloc indica al programa que es a partír de ese momento que se comienzan a rastrear las asignaciones de memoria de Python. Esto lo hace con el fin de que ya comenzando a rastrear estas asignaciones sea posible tomar la instantánea de estos bloques de memoria con la función take_snapshot() de esta librería.

Por otro lado la función stop() permite parar el rastreo de estas asignaciones de memoria en Python con lo cual limpia todos los rastros de asignaciones de memoria previamente recolectados. Para poder medir la cantidad de memoria se debe realizar otro take_snapshot() antes de stop(), ya que esta elimina el rastreo.

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

Para los resultados de la máqunia uno, se puede observar que el tiempo de ejecución del programa tiende a bajar conforme se aumenta el factor de carga. Se puede evidenciar en ambos MAPs.

Para los resultados de lo maquina dos, se puede observar que el tiempo de ejecución del programa baja conforme se aumenta el factor de carga. Esta tendencia se aplica para los dos métodos de colisiones.

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

En los resultados de la máquina uno, analizando los resultados sin considerar el esquema de colisiones es posible observar que el consumo de memoria es constante, esto se debe a que se crea un Map inicial cuyo tamaño M supera el valor máximo que predice el factor de carga que generá el mayor M (50/0.8), lo cual impide que se realice algún rehash y se gasta por ende la memoria indicada al inicio en la creación del Map.

Observando los resultados de la maquina dos, los cambios en el consumo de la memoria ocacionados por al cambio de factor de carga se ven casi nulos. Cuando se carga el catalogo con PROBING, no se observan cambio alguno en el consumo de memoria. Mientras que para la carga del catalogo como CHAINING, los cambios son muy pequeños como para considerar diferencias sustanciales. Esto normalmente no deberia suceder, sin embargo, se mide en conjunto la memoria que se ocupa cargando la lista completa de videos y el indice por categoría.

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.

En el tiempo de ejecución considerando el sistema de manejo de colisiones es posible observar diferencias en el tiempo de ejecución. Para el caso de esquema de colisiones PROBING a medida

que aumenta el factor de carga se ve que el tiempo de ejecución decrece estrictamente. Caso contrario al esquema de colisiones CHAINING en donde se ve un mínimo con un factor de carga 4 y vuelve a incrementar el tiempo con un factor de carga 6.

Por otro lado, en el esquema PROBING se evidencia que los tiempos varían en un rango más amplio que en el esquema CHAINING donde los valores del tiempo a medida que aumenta el factor de carga estan mucho más cerca.

Estas diferencias entre tiempos se pueden deber a las formas en las que estos dos Maps manejan las colisiones, el sistema Probing debe de buscar un espacio vacío en caso que la función de Hash sea la misma que otro ya puesto en esa posición lo que le hará recorrer el Map hasta encontrar un espacio vacío, contrario al sistema Chaining el cual busca si la función de Hash da un resultado lo guarda en un bucket y no busca un espacio en el Map vacío.

Se observa que para los resultados de la maquina dos, el esquema de colisiones probing tiende a presentar tiempos de ejecución menores que con el esquema de colisiones CHAINING. Sin embargo, el primer esquema anteriormente mencionado muestra que varia entre mas tiempos de ejecución según la varianza de su factor de carga. Mientras que, para el esquema CHAINING, no varia mucho el tiempo de ejecución, por lo que se considera a este mas exacto y consistente.

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.

En cuanto al consumo de memoria, para la máquina uno, en ambos esquemas de colisiones el consumo de memoria es constante a medida que incrementan los factores de carga, sin embargo es posible observar que el esquema de colisiones CHAINING consume más memoria (constante) que el sistema PROBING.

Esto se debe a que el sistema de Separate Chaining a parte de almacenar el Map guarda buckets en los cuales serán almacenados los valores cuya llave corresponda a la llave de dicha posición donde se ha de guardar.

Para los resultados de la maquina 2, se observa que el esquema de colisiones PROBING consume menos memoria que el otro esquema de colisiones, sin importar el factor de carga.