Preguntas laboratorio 7

a. ¿Por qué en la función getTime() se utiliza time.perf_counter() en vez de otras funciones como time.process_time()?

R/ Porque esta función permite devolver el tiempo en fracciones de segundos en fracciones cortas y con una mejor resolución.

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

R/ Al activar start() se empiezan a registrar las asignaciones de memoria, y el stop() finaliza el proceso para medir la memoria.

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

R/ Porque si se realiza una medición de forma paralela no va a ser tan precisa. Cuando se mide paralelamente, no se puede medir el tiempo de memoria de forma detallada, y si se mide desde el tiempo de memoria, sería un tiempo más corto, ya que contaría solo desde que empezó a cargar datos.

d. Teniendo en cuenta cada uno de los requerimientos del reto ¿Cuántos índices implementaría en el Reto? y ¿Por qué?

R/ Se pueden utilizar cuatro índices. Uno para que la llave sea ID, otro para que la llave sea el país, otro para que la llave sea el nombre de la compañía y otro para que la llave sea la ciudad. Elegimos estos ya que son los criterios que usualmente se piden en los requerimientos.

e. Según los índices propuestos ¿en qué caso usaría Linear Probing o Separate Chaining en estos índices? y ¿Por qué?

R/ Se utiliza linear probing porque es más útil para tamaños de mapa pequeños y baja carga de datos, con colisiones moderadas. Eficiente en espacio y acceso rápido.
Por otro lado Separate Chaining es más util para tamaños de mapa grandes o alta carga de datos, con muchas colisiones esperadas. Tiene un mejor rendimiento en escenarios de alta carga, aunque puede ocupar más espacio.

En el caso específico dado, dado que los mapas 'jobs_empresa' y 'jobs_ciudad' tienen un tamaño de 7000, es probable que Separate Chaining sea más adecuado. Para 'jobs_country' (tamaño 1000), si se espera una baja carga de datos, Linear Probing podría ser una opción, pero Separate Chaining podría dar más facilidad.

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

R/ Para probing necesitamos dividir el número de elementos por el factor de carga deseado.
3,257,008/ 0.1 = 32,570,080
3,257,008/0.5 = 6,514,016

3,257,008/0.7 = 4,652,868.57
3,257,008/0.9 = 3,618,897.78

En el caso de chaining el tamaño de la tabla hash es igual al número total de elementos, ya que cada celda de la tabla contendrá una lista enlazada de elementos que tienen la misma ubicación de dispersión.

g. ¿Qué cambios percibe en el tiempo de ejecución al modificar el factor de carga máximo para cargar las ofertas de trabajo?

R/ Para la carga de catálogos con el método de probing, se observa un aumento general en el tiempo de ejecución a medida que aumenta el factor de carga máximo de 0.1 a 0.9, pero la relación no es lineal y hay variaciones en el tiempo de ejecución debido a distintos factores como eficiencia del algoritmo.

En cuanto al método de chaining, al aumentar el factor de carga máximo de 2.00 a 8.00, también se registra un aumento en el tiempo de ejecución. Sin embargo, al igual que con el probing, la relación no es lineal. En ambos métodos, un factor de carga más alto muestra tiempos de ejecución más largos.

h. ¿Qué cambios percibe en el consumo de memoria al modificar el factor de carga máximo para cargar las ofertas de trabajo?

Al modificar el factor de carga máximo para cargar las ofertas de trabajo, se observa que el consumo de datos permanece prácticamente constante tanto en el método de probing como en el de chaining. Esto muestra que el factor de carga no tiene un impacto significativo en la cantidad de datos durante el proceso de carga de catálogos. Es importante mencionar que estos resultados pueden depender de otras variables.

i. **¿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/ Cambiar la forma en que manejamos las colisiones entre datos puede afectar cuánto tarda una tarea en completarse. Si usamos un método más complicado, podría llevar más tiempo porque necesita resolver las colisiones de manera más detallada. Pero si usamos un método más eficiente, el tiempo podría ser menor ya que se pueden evitar algunos problemas. La opción ideal depende del tipo de datos y de lo importante que sea la rapidez en la tarea que se está realizando.

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

R/ Al modificar el esquema de colisiones afecta la cantidad de memoria porque si usamos un método más simple, es probable que necesitemos menos memoria porque no requerirá tantas cosas adicionales para funcionar. Pero si el método es más complicado, necesitará más memoria porque necesita más herramientas para hacer su trabajo.

k. **¿Qué configuración de ideal ADT Map escogería para el índice de ofertas de trabajo por "id"?, especifique el mecanismo de colisión, el factor de carga y el número inicial de elementos.**

R/ Usaría Chaining porque da una forma eficiente de manejar las colisiones, ya que permite almacenar múltiples elementos con la misma clave en una lista asociada a esa clave. Además, un factor de carga de 2.0 sería adecuado porque puede mantener un buen rendimiento, pero sin consumir excesiva memoria. Por ultimo el número inicial de elementos sería 1000 para acomodar las ofertas de trabajo iniciales sin desperdiciar memoria.

Observaciones laboratorio 7

	Maquina 1	Maquina 2
Procesadores	Intel Core i5-1135G7	Intel Core 5
Memoria Ram (GB)	8	8
Sistema operativo	Windows 11	Windows 10

Carga Catalogo PROBING

Factor de Carga (PROBING)	Consumo de Datos [kB]	Tiempo de Ejecución Real @LP [ms]
0.1	394799.45	3145.43
0.5	394798.73	3241.76
0.7	394798.52	3197.89
0.9	394798.71	3321.28

Carga de Catálogo CHAINING

Factor de Carga (CHAINING)	Consumo de Datos [kB]	Tiempo de Ejecución Real @SC [ms]
2.00	215751.60	2684.67
4.00	233585.82	3187.09
6.00	233585.87	2864.99
8.00	233585.82	2894.89