LABORATORIO No. 7: Mecanismos de Colisión

Objetivos

Comprender la implementación del ADT Tabla de Símbolos no ordenadas (Map) y su uso para la solución de problemas. Al finalizar este laboratorio el estudiante estará en capacidad de:

1. Entender a usar las tablas de hash como estructura de datos y su impacto en los órdenes de crecimiento temporal.
2. Familiarizarse con las estrategias de manejo de colisiones en tablas de hash
3. Comparar los tiempos de respuesta de las tablas de hash cuando su factor de carga cambia

Fecha Límite de Entrega

Recuerde que para el final de la sesión del laboratorio los miembros del grupo deben completar hasta el paso **PASO 6: Completar cambios para medir memoria**

La entrega completa del laboratorio hasta el **PASO 16: Compartir resultados con los evaluadores** será el martes 11 de octubre antes de la media noche (11:59 p.m.).

Preparación del Laboratorio

Revisar el API del TAD Map ubicado en **"\DISClib\ADT\map.py"** y específicamente las estructuras de datos **chaininghashtable.py** y **probinghashtable.py** localizadas en la carpeta **"\DISClib\DataStructures".**

Se recomienda instalar las extensiones de **VS Code** para medir el consumo de recursos y explorar marcas ("TODO") dentro del código.

- **Todo Tree,** URL: Gruntfuggly.todo-tree
- **Resource Monitor,** URL: mutantdino.resourcemonitor

Trabajo Propuesto

PASO 1: Copiar el ejemplo en su organización

Copie/Haga **Fork** del repositorio del laboratorio en su organización con el procedimiento aprendido en las prácticas anteriores.

El repositorio del proyecto base que utiliza este laboratorio es el siguiente

- https://github.com/ISIS1225DEVS/ISIS1225-SampleCollision.git

Antes de clonar el repositorio en su computador diríjase a su organización (Ej.: ***EDA2021-1-SEC02-G01*** para el primer grupo de la sección 2 del curso) y cambie el nombre del repositorio de acuerdo con el esquema **LabCollision-S<<XX>>-G<<YY>>** donde **XX** es el número de la semana de la práctica y donde **YY** es el número del grupo de trabajo. (Ej.: **LabCollision-S07-G01** para este **séptimo laboratorio** hecho por el **grupo 1** de la **sección 2).**

Recuerde que **NO necesita** agregar la sección o el semestre en este nombre porque ya está identificado en su organización.

PASO 2: Descargar el ejemplo

Después de renombrar el proyecto dentro de su organización ya puede clonar el proyecto. Descargue el código en su máquina local **(git clone)** siguiendo lo aprendido en las practicas anteriores.

Recuerde modificar el **README** del repositorio para incluir los nombres de los integrantes del grupo.

PASO 3: Ejecutar y analizar el ejemplo

El proyecto **SampleCollision** busca ayudarnos a medir el comportamiento de las Tablas de Símbolos no ordenadas (Map), específicamente el tiempo de ejecución y la cantidad de memoria consumida.

Antes de iniciar a explorar y modificar el ejemplo, recuerde descargar los datos de trabajo **Goodreads** disponibles en el portal oficial del curso en BrightSpace. Descargue el **Zip,** descomprímalo y guarde los archivos CSV en la carpeta ***/Data/GoodReads/** de su copia local de código.

Diríjase al archivo **view.py** y ejecútelo, y seleccione secuencialmente la **opción 1** y **2** para iniciar el catálogo y cargar información respectivamente. Al momento de seleccionar la **opción 2** el programa le pedirá si desea medir la memoria utilizada, responda afirmativamente con "True" y observará la siguiente repuesta.

```
Seleccione una opción para continuar
2
Cargando información de los archivos ....
Desea observar el uso de memoria? (True/False)
Respuesta: True
Libros cargados: 10000
Autores cargados: 5841
Géneros cargados: 34252
Tiempo [ms]:  47177.267 || Memoria [kB]:  295834.467
Bienvenido
1- Inicializar Catálogo
2- Cargar información en el catálogo
3- Consultar los libros de un año
4- Consultar los libros de un autor
5- Consultar los Libros por etiqueta
6- Ordenar mejores libros de un año
0- Salir
Seleccione una opción para continuar
```

En caso de que responda con "False" podrá ver el siguiente comportamiento.

```
Seleccione una opción para continuar
2
Cargando información de los archivos ....
Desea observar el uso de memoria? (True/False)
Respuesta: False
Libros cargados: 10000
Autores cargados: 5841
Géneros cargados: 34252
Tiempo [ms]:  18545.247
Bienvenido
1- Inicializar Catálogo
2- Cargar información en el catálogo
3- Consultar los libros de un año
4- Consultar los libros de un autor
5- Consultar los Libros por etiqueta
6- Ordenar mejores libros de un año
0- Salir
Seleccione una opción para continuar
```

Note que el tiempo de ejecución varia cuando de aproximadamente **47 segundos** a **18 segundos** cuando se toma registro de la memoria utilizada, esto se debe a cómo interactúa la librería **tracemalloc** con la estructura del código.

El tiempo realista de ejecución de la operación es **18.54 segundos,** mientras que los **47.2 segundos** reportados junto a los **295.8 MB** se deben la función para medir la memoria utilizada guarda una copia de todas las operaciones realizadas para luego calcular el espacio utilizado.

Ahora, ejecutando la **opción 3** y seleccionando si se desea recopilar la información sobre la memoria utilizada o no("True/False"), podemos ver un comportamiento similar donde al medir la memoria se aumenta el tiempo de ejecución como se ve en las imágenes a continuación.

```
Seleccione una opción para continuar
3
Buscando libros del año?: 2011
Desea observar el uso de memoria? (True/False)
Respuesta: T
Se encontraron: 557 Libros.
Tiempo [ms]:  0.158 || Memoria [kB]:  0.594
Imprimiendo los primeros 3 y ultimos libros...
Titulo: Divergent (Divergent, #1) ISBN: 62024035 Rating: 4.24
Titulo: Fifty Shades of Grey (Fifty Shades, #1) ISBN: 1612130291 Rating: 3.67
Titulo: Fahrenheit 451 ISBN: 307347974 Rating: 3.97
Titulo: The Virgin Cure ISBN: 676979564 Rating: 3.74
Titulo: The Tycoon's Revenge (Baby for the Billionaire, #1) ISBN:  Rating: 3.87
Titulo: Cinderella Ate My Daughter: Dispatches from the Frontlines of the New Girlie-Girl Culture ISBN: 61711527 Rating: 3.65
```

```
Seleccione una opción para continuar
3
Buscando libros del año?: 2011
Desea observar el uso de memoria? (True/False)
Respuesta: F
Se encontraron: 557 Libros.
Tiempo [ms]:  0.019
Imprimiendo los primeros 3 y ultimos libros...
Titulo: Divergent (Divergent, #1) ISBN: 62024035 Rating: 4.24
Titulo: Fifty Shades of Grey (Fifty Shades, #1) ISBN: 1612130291 Rating: 3.67
Titulo: Fahrenheit 451 ISBN: 307347974 Rating: 3.97
Titulo: The Virgin Cure ISBN: 676979564 Rating: 3.74
Titulo: The Tycoon's Revenge (Baby for the Billionaire, #1) ISBN:  Rating: 3.87
Titulo: Cinderella Ate My Daughter: Dispatches from the Frontlines of the New Girlie-Girl Culture ISBN: 61711527 Rating: 3.65
```

Esto quiere decir que al medir la memoria utilizada en un algoritmo estamos alterando su tiempo de ejecución y que la respuesta real del mismo **NO** es el que se observa junto con la medida de memoria. Sino que el tiempo **real** de procesamiento es el medido sin tener en cuenta la memoria.

Entonces, tomando los ejemplos previos ejecutando las **opciones 1, 2 y 3,** el tiempo de ejecución y la memoria utilizada para la **carga de datos** es de **18545 ms** y **295834 kB** respectivamente. Y el tiempo más la memoria para **consultar el índice de libros por año** es de **0.019 ms** y **0.594 kB**

Para estudiar detalladamente cómo se implementaron estas opciones diríjase al **model.py** a las funciones **addBook(), addTag(), addBookTag()** y **getBooksByYear()** donde encontrara el código documentado de como guardar en memoria la información utilizando los **TAD List** y **TAD Map** del curso.

Ahora, para entender como medir el tiempo de ejecución y la memoria utilizada por las funcionalidades diríjase al **controller.py**, allí encontrará cuatro (4) funciones básicas para esta tarea, cuyos nombres son: **getTime(), getMemory(), deltaTime()** y **deltaMemory().** Para entenderlas mejor se describe a continuación cada una de estas.

Primero, la función **getTime()** toma el tiempo del reloj del procesador utilizando la librería **"time"** y devuelve su resultado en milisegundos [ms] como flotante.

```
def getTime():
    """
    devuelve el instante tiempo de procesamiento en milisegundos
    """
    return float(time.perf_counter()*1000)
```

Segundo, la función **deltaTime()** simplemente calcula la diferencia del tiempo entre los dos instantes diferentes tomados con la función **getTime().**

```
def deltaTime(end, start):
    """
    devuelve la diferencia entre tiempos de procesamiento muestreados
    """
    elapsed = float(end - start)
    return elapsed
```

Tercero, la función **getMemory()** utiliza la librería nativa **"tracemalloc"** de Python para tomar una imagen de la memoria asignada por el programa en un instante de tiempo.

```
def getMemory():
    """
    toma una muestra de la memoria alocada en instante de tiempo
    """
    return tracemalloc.take_snapshot()
```

por último, **deltaMemory()** utiliza **"tracemalloc",** con dos muestras de memoria (resultados de invocar **getMemory()**) y estima la memoria asignada por Python calculando las diferencias entre las dos muestras como se puede ver a continuación.

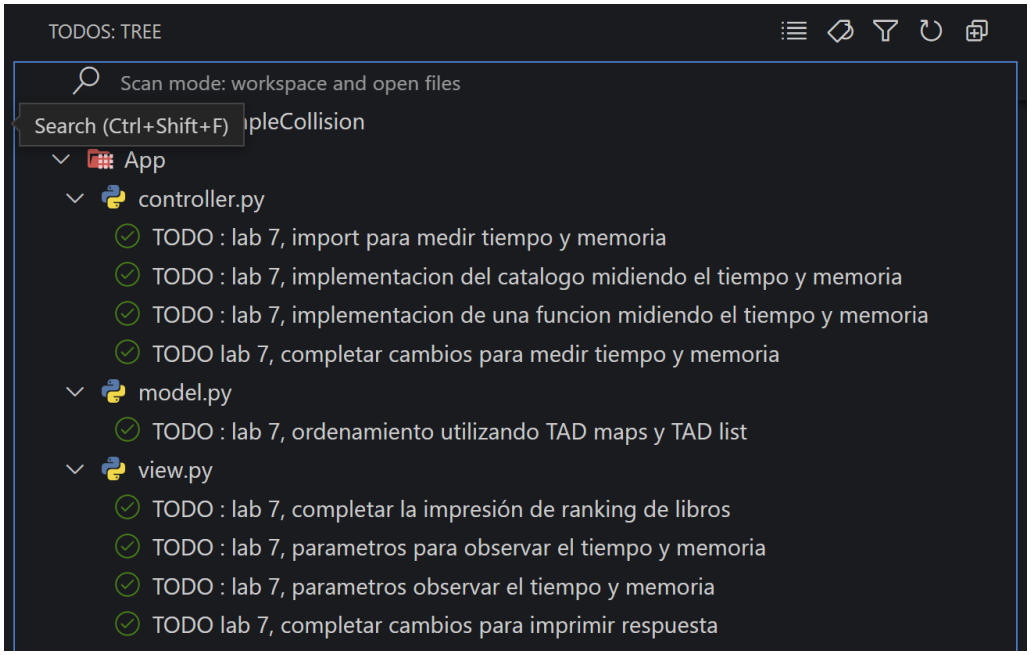
```python
def deltaMemory(start_memory, stop_memory):
    """
    calcula la diferencia en memoria alocada del programa entre dos
    instantes de tiempo y devuelve el resultado en kBytes (ej.: 2100.0 kB)
    """
    memory_diff = stop_memory.compare_to(start_memory, "filename")
    delta_memory = 0.0

    # suma de las diferencias en uso de memoria
    for stat in memory_diff:
        delta_memory = delta_memory + stat.size_diff
    # de Byte -> kByte
    delta_memory = delta_memory/1024.0
    return delta_memory
```

Estas cuatro funciones (**getTime(), deltaTime(), getMemory()** y **deltaMemory()**) se deben incluir en el uso de funciones como **getBooksByYear()** y **sortBooksByYear()** a nivel del controlador para calcular el tiempo de ejecución y la cantidad de memoria utilizada.

Dentro de la implementación de las funciones en el controlador es importante recordar que la librería de **"tracemalloc"** debe iniciar el proceso de toma de datos antes de ejecutar la función del **model.py** con el comando **start()** y debe finalizar subsecuentemente su proceso con **stop().**

PASO 4: Estudiar modificaciones del código

Para entender bien como se ejecutan las mediciones en las **opciones 2 y 3** e integrar los cambios a un próximo código funcional exploraremos el proyecto con la extensión **Todo Tree** de **VS Code** para localizar las notas *"TODO"* dentro de la implementación como se ve a continuación.



Primero diríjase al **controller.py** y examine los **tres (3) primeros** cambios con las marcas **"TODO".** En la primera anotación se incluyen las importaciones de las librerías **"time"** y **"tracemalloc"** como se ve en el siguiente segmento de código.

```
# TODO: import para medir tiempo y memoria
import time
import tracemalloc
import config as cf
import model
import csv
```

La segunda anotación resalta que la función **loadData()** ya incluye las modificaciones para medir tiempo de ejecución y memoria utilizada manejando las librerías **time** y **tracemalloc** como se muestra a continuación.

```
def loadData(ctrlr, memflag=True):
    """
    Carga los datos de los archivos y cargar los datos en la
    estructura de datos
    """
    # TODO: lab 7, implementacion del catálogo midiendo el tiempo y memoria
    # toma el tiempo al inicio del proceso
    start_time = getTime()

    # inicializa el proceso para medir memoria
    if memflag is True:
        tracemalloc.start()
        start_memory = getMemory()

    loadBooks(ctrlr)
    loadTags(ctrlr)
    loadBooksTags(ctrlr)

    # toma el tiempo al final del proceso
    stop_time = getTime()
    # calculando la diferencia en tiempo
    delta_time = deltaTime(stop_time, start_time)

    # finaliza el proceso para medir memoria
    if memflag is True:
        stop_memory = getMemory()
        tracemalloc.stop()
        # calcula la diferencia de memoria
        delta_memory = deltaMemory(stop_memory, start_memory)
        # respuesta con los datos de tiempo y memoria
        return delta_time, delta_memory

    else:
        # respuesta sin medir memoria
        return delta_time
```

La función **loadData()** tiene un parámetro opcional de entrada denominado **memflag** el cual se utiliza para parametrizar si se desea medir la memoria utilizada o no.

Recuerde que la función **getBooksYear()** también incluye modificaciones similares al **loadData()** para medir tiempo de ejecución y la memoria utilizada.

Por último, vaya al archivo **view.py** y examine las **dos (2)** primeras modificaciones.

La **primera marca** dentro del código para la **opción 2** permite elegir si se mide o no la memoria utilizada e imprimir e imprime por consola los resultados de la carga del catálogo utilizando la función de **printLoadDataAnswer()** para desplegar los resultados de la operación como se ve a continuación.

```
    elif int(inputs[0]) == 2:
        # TODO: modificaciones para observar el tiempo y memoria
        print("Cargando información de los archivos ....")
        print("Desea observar el uso de memoria? (True/False)")
        mem = input("Respuesta: ")
        mem = castBoolean(mem)
        print(mem)
        answer = controller.loadData(ctrlr, memflag=mem)
        print('Libros cargados: ' + str(controller.booksSize(ctrlr)))
        print('Autores cargados: ' + str(controller.authorsSize(ctrlr)))
        print('Géneros cargados: ' + str(controller.tagsSize(ctrlr)))
        printLoadDataAnswer(answer)
```

Al observar en detalle la función **printLoadDataAnswer()**, podemos observar que la impresión de resultados depende de la configuración de entrada (medir memoria utilizada o no) y que por lo tanto hay un flujo de control inmerso en las operaciones de despliegue de resultados en el **view.py** como se aprecia en el siguiente bloque de código.

```
def printBooksbyYear(answer):
    """
    Imprime los libros que han sido publicados en un
    año
    """
    if isinstance(answer, (list, tuple)) is True:
        if len(answer) == 2:
            books = answer[0]
            time = answer[1]
            print('Se encontraron: ' + str(lt.size(books)) + ' Libros.')
            print("Tiempo [ms]: ", f"{time:.3f}")
        elif len(answer) == 3:
            books = answer[0]
            time = answer[1]
            memory = answer[2]
            print('Se encontraron: ' + str(lt.size(books)) + ' Libros.')
            print('Tiempo [ms]: ', f"{time:.3f}", '||',
                  'Memoria [kB]: ', f"{memory:.3f}")
        print("Imprimiendo los 3 primeros y 3 ultimos libros...")
        i = 1
        for book in lt.iterator(books):
            if i <= 3 or i > lt.size(books) - 3:
                print('Titulo:', book['title'],
                      'ISBN:', book['isbn'],
                      'Rating:', book['average_rating'])
            i += 1
        print("\n")
    else:
        print("No se encontraron libros.\n")
```

una de las cosas mas importantes de esta funcion es que los resultados del tiempo de ejecución y la memoria utilizada se aproximan a la tercera cifra decimal para obtener datos con una precisión adecuada como se observa en la línea de código *"print('Tiempo [ms]: ', f"{time:.3f}", '||',..."*.

Recordemos que la **segunda marca "TODO"** para la implementación de la **opción 3 que** también podemos apreciar las mismas modificaciones para imprimir adecuadamente el resultado de la consulta del índice anual según lo defina el usuario.

Esto finaliza la exploración del código MVC y ahora puede apreciar como las opciones 2 y 3 del menú permiten recuperar los resultados del tiempo de ejecución y la memoria utilizados en las operaciones.

Ahora, considere responder las siguientes preguntas en el documento **Observaciones-Lab 7.docx:**

a) ¿Por qué en la función **getTime()** se utiliza **time.perf_counter()** en vez de otras funciones como **time.process_time()**?
b) ¿Por qué son importantes las funciones **start()** y **stop()** de la librería **tracemalloc**?
c) ¿Por qué no se puede medir paralelamente el **uso de memoria** y el **tiempo de ejecución** de las operaciones?

PASO 6: Completar cambios para medir memoria

Ahora, aplique las modificaciones estudiadas sobre la **opción 6.** Para ello utilice la extensión **Todo Tree** de **VS Code** y localice las notas *"TODO"* con el mensaje *"lab 7, completar..."*.

Primero diríjase al **view.py** y modifique las dos (2) marcas *"TODO"* como se muestra en los siguientes bloques de código para la función **printBestBooks()** y la ejecución de la **opción 6** respectivamente.

```
def printBestBooks(books):
    """
    Imprime la información de los mejores libros
    por promedio
    """
    # TODO: lab 7, completar la impresión de ranking de libros
    if isinstance(answer, (list, tuple)) is True:
        if len(answer) == 2:
            books = answer[0]
            time = answer[1]
            print("Tiempo [ms]: ", f"{time:.3f}")
        elif len(answer) == 3:
            books = answer[0]
            time = answer[1]
            memory = answer[2]
            print('Tiempo [ms]: ', f"{time:.3f}", '||',
                  'Memoria [kB]: ', f"{memory:.3f}")
        size = lt.size(books)
        if size:
            print(' Estos son los mejores libros: ')
            for book in lt.iterator(books):
                print('Titulo:', book['title'],
                      'ISBN:', book['isbn'],
                      'Rating:', book['average_rating'])
            print("\n")
    else:
        print("No se encontraron libros.\n")
```

```
    elif int(inputs[0]) == 6:
        number = input("Buscando libros del año?: ")
        rank = input("Cuantos libros en el escalafón? (mayor a 0): ")
        number = int(number)
        rank = int(rank)
        print("Desea observar el uso de memoria? (True/False)")
        mem = input("Respuesta: ")
        mem = castBoolean(mem)
        # TODO lab 7, completar cambios para imprimir respuesta
        answer = controller.sortBooksByYear(ctrlr, number, rank, memflag=mem)
        printBestBooks(answer)
```

Ahora, diríjase al **controller.py** en la función **sortBooksByYear()** y complete el código de la siguiente manera:

```
def sortBooksByYear(ctrlr, year, rank, memflag=True):
    """
    Retorna los libros que fueron publicados
    en un año ordenados por rating
    """
    # TODO lab 7, completar cambios para medir tiempo y memoria
    # respuesta de la funcion
    books = None

    # toma el tiempo al inicio del proceso
    start_time = getTime()

    # inicializa el proceso para medir memoria
    if memflag is True:
        tracemalloc.start()
        start_memory = getMemory()

    # ejecutando funcion a medir
    books = model.sortBooksByYear(ctrlr['model'], year, rank)

    # toma el tiempo al final del proceso
    stop_time = getTime()
    # calculando la diferencia en tiempo
    delta_time = deltaTime(stop_time, start_time)

    # finaliza el proceso para medir memoria
    if memflag is True:
        stop_memory = getMemory()
        tracemalloc.stop()
        # calcula la diferencia de memoria
        delta_memory = deltaMemory(stop_memory, start_memory)
        # respuesta con los datos de tiempo y memoria
        return books, delta_time, delta_memory

    else:
        # respuesta sin medir memoria
        return books, delta_time
```

Al finalizar estos cambios y reinicie la aplicación la **opción 6** del menú debe ejecutar sin problemas.

Finalmente, asegúrese de hacer **commit** y **push** de los cambios en el repositorio con el comentario *"modificaciones de tiempos y memorias – Laboratorio 7"*.

Factores de Carga

Ahora que se entiende el uso del TAD Map, los factores de carga y cómo podemos medir la memoria que utilizan las tablas de hash en la carga de datos. Aplicaremos todo esto implementando el reto.

PASO 7: Copiar la plantilla del Reto No. 2 en su organización

Copie/Haga **Fork** del repositorio del reto en su organización con el procedimiento aprendido previamente. El repositorio del proyecto base que utiliza este módulo es:

https://github.com/ISIS1225DEVS/Reto2-Template

Antes de clonar el repositorio diríjase a su organización (Ej.: ***EDA2021-1-SEC02-G01*** para el primer grupo de la sección 2 del curso) y cambie el nombre del repositorio según el esquema **Reto2-G<<XX>>** donde **XX** es el número del grupo de trabajo. (Ej.: **Reto2-G01** para el **grupo 1** de la **sección 2).**

Recuerde que el repositorio **debe ser privado** y **NO necesita** agregar la sección o el semestre en este nombre.

PASO 8: Descargue el Reto No. 2

Después de renombrar el proyecto dentro de su organización ya puede clonar el proyecto. Descargue el código en su máquina local siguiendo lo aprendido en las practicas anteriores.

Recuerde modificar el archivo **README** del repositorio para incluir los nombres de los integrantes del grupo e identificar claramente el miembro quien implementara cada requerimiento individual. Un ejemplo es:

- *Req. 3 - Santiago Arteaga, 200411086, sa-artea@uniandes.edu.co*
- *Req. 4 - Carlos Lozano, 200211089, calozanog@uniandes.edu.co*
- *Req 5 – Fernando De la Rosa, 200015053, fde@uniandes.edu.co*

PASO 10: Aplicar ADT Map en el Reto No. 2

Tome como base el código implementado para resolver el Reto 1 (archivos **`view.py`, `controller.py`,** y **`model.py`**) y utilícelo para iniciar a implementar Reto 2.

Específicamente en el archivo **`model.py`** del reto 2 realice las siguientes modificaciones:

1) Agregue las importaciones apropiadas para utilizar ADT Map en el **`model.py`** de su reto.
2) Modifique su catálogo para que al cargar el contenido se cree un índice por géneros según la información de la columna **"listed_in"** en el archivo CSV utilizando la librería **Maps.py** de **DISClib.**

PASO 11: Actualizar el repositorio en la rama principal

Confirme los cambios con los comandos **`Commit`** y **`Push`** en la rama ***main*** local y de GitHub con el comentario *"Primer avance – Reto 2"* antes de la fecha límite de entrega.

PASO 12: Preparar las pruebas

Antes de iniciar las pruebas sobre la implementación de su Reto No. 2 recomendamos tener en cuenta las siguientes instrucciones:

1) Cada estudiante debe ejecutar las pruebas propuestas en sus máquinas.
2) Para la toma de datos se debe utilizar el archivo de mayor tamaño que los computadores de todos los integrantes del equipo pueda procesar. Por decir, si todos los integrantes pueden cargar el archivo **large** deben usar este tamaño para la toma de datos. Mientras que si el computador de un estudiante solo puede cargar el **50pct** entonces todos los estudiantes deberán utilizar este subconjunto.
3) Diligenciar la Tabla 1 en el documento **Observaciones-lab7.docx** con la información de las máquinas de cómputo donde cada estudiante ejecutará las pruebas de carga de datos.

	Máquina 1	Máquina 2	Máquina 3
Procesadores			
Memoria RAM (GB)			
Sistema Operativo			

Tabla 1. Especificaciones de las máquinas para ejecutar las pruebas de carga de datos.

Por ejemplo:

	Máquina 1	Máquina 2	Máquina 3
Procesadores	Intel(R) Core (TM) i7-55000 CPU @2.40GHz 2.40GHZ	Intel(R) Core (TM) i7-10750H CPU @ 2.60GHz 2.59 GHz	Intel(R) Core (TM) i2400H CPU @ 2.2GHz 2.60 GHz
Memoria RAM (GB)	16.0 GBGB	32.0 GB	8.0 GB
Sistema Operativo	Windows 10 Pro-64 bits	Windows 10 Pro-64 bits	Windows 10 Pro-64 bits

4) Ejecute todas las pruebas de rendimiento propuestas en la misma máquina que reporto.
5) Cierre todas las aplicaciones que puedan afectar la toma de datos y sean **innecesarias** de su máquina, ej.: **Discord**, **OneDrive**, **Dropbox**, **Word** y en especial el navegador de internet (**Edge**, **Chrome**, **Firefox**, **Safari, ...**).
6) Se recomienda que cada prueba se ejecute por lo menos tres veces para poder obtener un promedio o consolidado consistente de las pruebas. Ej. Para un experimento para cargar el catálogo con factor de carga 0.5 se debe ejecutar por lo menos 3 veces (ideal 5 veces).
7) evitando errores dentro de la máquina como lo son actualizaciones o escaneos de seguridad, para registrar resultados adecuados en las tablas provistas.
8) Cada uno de los tiempos debe estas registrado en **milisegundos (ms)** y con 2 cifras decimales redondeado hacia arriba desde la mitad. Ej.: **43494.498587 ms** se aproxima a **43494.50 ms.**

PASO 13: Medir la carga de datos del Reto No. 2

En esta sección evaluaremos el efecto en el uso de la memoria y tiempo de ejecución para **TAD Map** en la implementación del reto.

Para ello, integre al código del Reto No. 2 los cambios estudiados en la primera parte de la práctica. Inclúyalos para medir el tiempo de ejecución y la memoria utilizada en la carga del catálogo de videos, recuerde que en la práctica anterior creo un índice por categorías con **TAD Map.**

Para ello siga las siguientes indicaciones:

1) Incluir las importaciones para medir tiempo y memoria en el **`controller.py.`**
2) Integrar las funciones **`getTime(), deltaTime(), getMemory()`** y **`deltaMemory()`** en el **`controller.py`**
3) Modificar la su función de carga del tamaño escogido para medir el tiempo de ejecución y la memoria utilizada en el proceso.

Al completar las modificaciones del Reto No. 2, diligencie la Tabla 2 y la Tabla 3 variando el factor de carga y los mecanismos de colisiones entre **"PROBING"** y **"CHAINING"** para la Tabla de Hash en el índice de géneros previamente implementado (e.j.: *`maptype='PROBING'`* y *`maptype='CHAINING'`*).

Carga de Catálogo PROBING

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

Tabla 2. Mediciones de tiempo y datos para diferentes factores de carga en PROBING.

Carga de Catálogo CHAINING

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

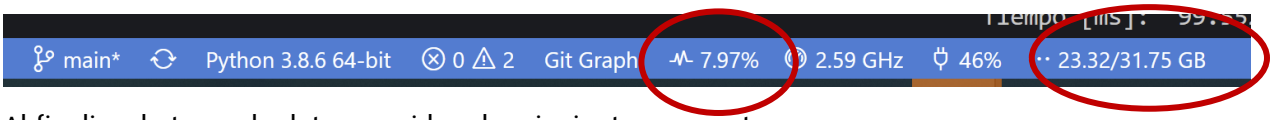
Tabla 3. Mediciones de tiempo y datos para diferentes factores de carga en CHAINING.

Recuerde que el tiempo de ejecución real es el que no se toma en paralelo con la memoria utilizada.

Durante las pruebas recomendamos escoger el archivo de datos más apropiado a las capacidades de sus máquinas de trabajo. Y utilizar el **Resource Monitor** de **VS Code** para monitorear la memoria disponible de su computador.

Recuerde que las pruebas tienen un **máximo de duración de 10.0 minutos,** en caso de que esto suceda aborte el procesamiento y reduzca la cantidad de datos que su aplicación planea manejar.

también aborte la prueba en caso de que la memoria se acerque a su capacidad máxima o cuando su procesador este sobre cargado por mucho tiempo. Ambos estados se pueden ver en la barra inferior de información de **VS Code** la extensión **Resource Monitor** como se muestra a continuación.



Al finalizar la toma de datos considere las siguientes preguntas:

d) Teniendo en cuenta cada uno de los requerimientos del reto ¿Cuántos índices implementaría en el Reto? y ¿Por qué?
e) Según los índices propuestos ¿en qué caso usaría **Linear Probing** o **Separate Chaining** en estos índices? y ¿Por qué?
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?
g) ¿Qué cambios percibe en el **tiempo de ejecución** al modificar el factor de carga máximo para cargar el catálogo de contenido Streaming?
h) ¿Qué cambios percibe en el **consumo de memoria** al modificar el factor de carga máximo para cargar el catálogo de contenido Streaming?
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.
j) ¿Qué cambios percibe en el **consumo de memoria** al modificar el esquema de colisiones?, si los percibe, describa las diferencias y argumente su respuesta.
k) ¿Qué configuración de ideal ADT Map escogería para el **índice géneros de contenido ("listed_in")**?, especifique el mecanismo de colisión, el factor de carga y el numero inicial de elementos.

Las preguntas de análisis y las tablas de datos deben estar en el documento **observaciones-lab7.docx,** al completarlas guarde una copia de este documento en formato **PDF** para evitar problemas de lectura en la evaluación y cópielo en el repositorio de la práctica.

PASO 14: Actualizar los repositorios

Para el repositorio del laboratorio confirme los cambios con los comandos **`Commit`** y **`Push`** en la rama **main** local y de GitHub con el comentario *"laboratorio 7 - Entrega final"* antes de la fecha límite de entrega.

Para el repositorio del Reto confirme los cambios con los comandos **`Commit`** y **`Push`** en la rama **main** local y de GitHub con el comentario *"Reto 2 – Avance en índices"* antes de la fecha límite de entrega.

PASO 15: Revisar entregables de la practica

Finalmente, para realizar la entrega del laboratorio revise que sus entregables de la practica estén completos. Para ello, siga las siguientes indicaciones:

1. Acceso a la organización GitHub del grupo para su profesor y monitores de la sección de laboratorio.
2. **README** de ambos repositorios con los datos completos de los integrantes del grupo (nombre completo, correo Uniandes y código de estudiante).

3. Enlace al repositorio GitHub **Reto2-G<<XX>>** con rama ***Main*** actualizada con el comentario *"Reto 2 – Avance en índices"* antes del límite de entrega.
4. Enlace al repositorio GitHub **LabCollision-S<<XX>>-G<<YY>>** con rama ***Main*** actualizada con el comentario *"Laboratorio 7 – Entrega final"* antes del límite de entrega.
5. Incluir en repositorio del laboratorio en la carpeta **Docs** el documento **observaciones-lab7.pdf** con las respuestas a las preguntas de observación.
    a. Los datos reportados en la Tabla 1, Tabla 2 y Tabla 3.
    b. Las 2 graficas comparativas generada por los resultados de las pruebas de carga de datos.
    c. Las respuestas a las preguntas de observación:
        i. ¿Por qué en la función **getTime()** se utiliza **time.perf_counter()** en vez de otras funciones como **time.process_time()**?
        ii. ¿Por qué son importantes las funciones **start()** y **stop()** de la librería **tracemalloc**?
        iii. ¿Por qué no se puede medir paralelamente el **uso de memoria** y el **tiempo de ejecución** de las operaciones?
        iv. Teniendo en cuenta cada uno de los requerimientos del reto ¿Cuántos índices implementaría en el Reto? y ¿Por qué?
        v. Según los índices propuestos ¿en qué caso usaría **Linear Probing** o **Separate Chaining** en estos índices? y ¿Por qué?
        vi. 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?
        vii. ¿Qué cambios percibe en el **tiempo de ejecución** al modificar el factor de carga máximo para cargar el catálogo de contenido Streaming?
        viii. ¿Qué cambios percibe en el **consumo de memoria** al modificar el factor de carga máximo para cargar el catálogo de contenido Streaming?
        ix. ¿Qué cambios percibe en el **tiempo de ejecución** al modificar el esquema de colisiones?, si los percibe, describa las diferencias y argumente su respuesta.
        x. ¿Qué cambios percibe en el **consumo de memoria** al modificar el esquema de colisiones?, si los percibe, describa las diferencias y argumente su respuesta.
        xi. ¿Qué configuración de ideal ADT Map escogería para el **índice géneros de contenido ("listed_in")**?, especifique el mecanismo de colisión, el factor de carga y el numero inicial de elementos.

PASO 16: Compartir resultados con los evaluadores

Envíe los **enlaces (URL)** de los dos repositorios por **BrightSpace** antes de la fecha límite de entrega.

Recuerden que cualquier documento solicitado durante en la práctica debe incluirse dentro del repositorio GIT y que solo se calificaran los entregables hasta el último ***COMMIT*** realizado antes de la media noche **(11:59 PM) del 11 de octubre de 2022.**