OBSERVACIONES DE LA PRACTICA

Ana Sofía Castellanos- Código: 202114167
Martín Santiago Galván Castro 201911013

1) ¿Qué estructura de datos se usa para este índice?
Este índice por autor hace uso de la estructura de datos tabla de Hash con un método de colisiones del tipo Separate Chaining.

2) ¿Cuántos elementos se espera almacenar inicialmente?
En la línea de código se especifica en el primer elemento, que se espera almacenar inicialmente 800 autores.

3) ¿Cuál es el factor de carga?
El factor de carga N/M es de 4.0

4) ¿Con cuántos elementos serán necesarios agregar para hacer re-hash de la tabla?
Considerando que se tiene un factor de carga igual a 4 y que el tamaño original de la lista es M = 800. Usando la ecuación de factor de carga se obtiene que:

$$N/M = 4$$
$$N/800 = 4$$
$$N = 3200$$

5) ¿Qué hace la instrucción "**mp.put(...)**"?
La función mp.put(…) añade al mapa que ingresa por parámetro la llave y el valor que ingresa por parámetro. Para este caso añade al mapa bookIds que se encuentra en el catálogo, como llave el id de goodreads del libro y como valor el libro.

Esta función llama a la función put archivo map.py presente en la carpeta ADT de DISClib, esta función a su vez llama a la función put presente en el archivo mapstructure de la carpeta DataStructures. En esta función se revisa el tipo de tabla de hash que se va a implementar, si es chaining llama a la función put del archivo chaininghashtable, de lo contrario llama a la función put de archivo probehashtable.

Para el caso de la tabla de hash del tipo chaining, se obtiene un número de la función de hash del mapa que entra por parámetro y la llave que ingresa por parámetro. Luego se obtiene el bucket (lista) del map a partir del número hash a través de getElement(map['table'], hash). A partir de allí se crea una nueva entrada del mapa con la llave y valor ingresados por parámetro y se revisa si esta entrada ya existe en el bucket. De existir se reemplazan los valores que estaban en ese bucket, si no está se añade este nuevo elemento al final del bucket y se actualiza la cantidad de elementos del map y el factor de carga. Finalmente se revisa si el factor de carga supera el límite, de ser así se hace rehash.

Para el caso de la tabla de hash del tipo probing se obtiene un número de la función de hash del map que ingresa por parámetro y la llave que ingresa por parámetro. Luego se crea una nueva entrada a partir de la llave y el valor ingresados por parámetro y se halla la posición en la que

se ubicará dicha entrada dependiendo de si la posición del hash corresponde a la llave ingresada o si está en None. A partir de allí se cambia la información que se encuentra en esa posición hallada por la que ingresa por parámetro y se actualiza la cantidad de elementos del map y el factor de carga. Finalmente se revisa si el factor de carga supera el límite, de ser así se hace rehash.

6) ¿Qué papel cumple **"book['goodreads_book_id']"** en esa instrucción?
En el caso de la línea de código "mp´.put(catalog['bookIds'], book['goodreads_book_id'], book) ". La parte cumple la función de ser la llave del libro. Esto se hace de esta manera ya que en catalogo['booksIds'] se guardan los libros conforme a su id.

7) ¿Qué papel cumple ***"book"*** en esa instrucción?
El ultimo input de la función mp.put(…) hace referencia al valor que se guarda en la pareja llave valor. En este caso, "book" o un libro en específico es el que se guarda como valor, en donde su llave es su id.

8) ¿Qué hace la instrucción **"mp.get(…)"**?
La función mp.get(…) retorna del mapa 'years' del catálogo la pareja llave valor que cumple que su llave es igual a la llave que ingresa por parámetro.

Esta función llama a la función get del archivo map.py presente en la carpeta ADT de DISClib, esta función a su vez llama a la función get presente en el archivo mapstructure de la carpeta DataStructures. En esta función se revisa el tipo de tabla de hash que se va a implementar, si es chaining llama a la función get del archivo chaininghashtable, de lo contrario llama a la función get de archivo probehashtable.

Para el caso de la tabla de hash del tipo chaining se obtiene un número de la función de hash del map que ingresa por parámetro y la llave que ingresa por parámetro. Se obtiene el bucket donde se encuentra el elemento a través de getElement(map['table'], hash) y se verifica si el elemento está en el bucket a través de la función isPresent(bucket, key) de estar devuelve el elemento con getElement(…) de lo contrario devuelve None.

Para el caso de la tabla de hash del tipo probing, se obtiene un número de la función de hash del mapa que entra por parámetro y la llave que ingresa por parámetro. Luego se revisa si existe este elemento en el mapa usando la función findSlot(…). A partir de allí se revisa si existe y devuelve el elemento de esta posición, en caso de no estar se devuelve None.

9) ¿Qué papel cumple ***"year"*** en esa instrucción?
"year" es el año de consulta. En este caso, para la parte de mp.get(…) revisa en el índice de años si existe una pareja llave-valor, cuya llave sea "year". Esto con el objetivo de obtener dicha pareja y extraer los libros de esta. En caso de que no exista devuelve None ya que no hay libros de dicho año.

10) ¿Qué hace la instrucción **"me.getValue(…)"**?
La función me.getValue(…) retorna a partir de una pareja llave valor ingresada por parámetro el valor de dicha pareja. En este caso específico a partir de la pareja key = año_publicacion y

value = {year:…, books: lista} se obtiene el valor {year:…, books: lista con los libros de dicho año}.

Esta función se importa de la librería DataStructures que se encuentra en ADT. Mas específicamente, se encuentra en el archivo mapentry.py. MapEntry representa una entrada, es decir, una pareja llave-valor en un Map. Esta librería contiene funciones para realizar operaciones relacionadas a la pareja de llaves. La función getValue(…) recibe como entrada una entrada de un mapa, es decir una pareja llave-valor. Y devuelve el valor de dicha pareja.