Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI-2692 - Laboratorio de Algoritmos y Estructuras de Datos II
Trimestre Enero-Marzo 2020

Implementación de una Tabla de Hash usando el método de encadenamiento

1. Descripción de las actividades a realizar

El objetivo del laboratorio es realizar la implementación de la estructura de datos Tabla de Hash que usa el método de encadenamiento para la resolución de colisiones. Usted debe implementar una Tabla de Hash en donde las claves son de tipo entero, y el dato a almacenar que esta asociado a cada clave, es un elemento de tipo String. El encadenamiento debe ser llevado a cabo mediante una lista doblemente enlazada. La figura 1 muestra un ejemplo de la Tabla de Hash, la cual es semejante a la que se presenta en el Cormen et al [1].

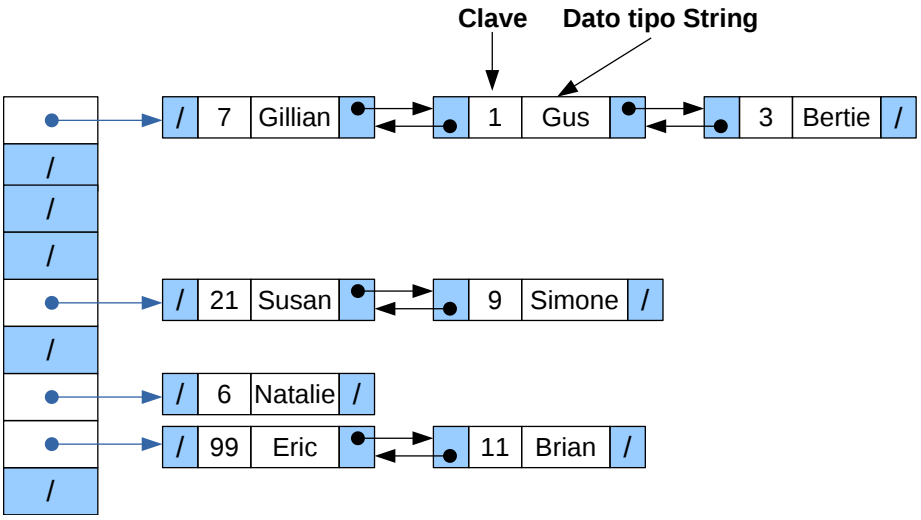

Figura 1: Ejemplo de la Tabla de Hash donde el encadenamiento es realizado con una lista doblemente enlazada. Las claves son elementos de tipo entero, y el dato asociado a cada clave es de tipo String.

Se quiere implementar una Tabla de Hash estática. Esto es, la tabla se crea con un número de n *slots* y el número de *slots* no cambia sin importar el número de elementos que se agregan a la tabla. La lista doblemente enlazada estará constituida por elementos de tipo *HashEntry*. En la figura 1 podemos observar que el tipo *HashEntry* debe contener al menos cuatro campos: uno para la clave, uno para el dato tipo String y dos para las referencias a otros dos elementos de tipo *HashEntry*. El tipo *HashEntry* debe ser implementado como una clase pública de Python, cuyo constructor recibe como entrada un clave y un String, esto es:

crearHashEntry(int *clave*, String *dato*)

La Tabla de Hash debe ser implementada como una clase pública de Python, y debe tener las siguientes operaciones:

Crear tabla: Se crea una tabla con un tamaño inicial de n *slots*.

crearTabla(int n)

Agregar elemento: Se agrega un elemento e de tipo *HashEntry* a la tabla de hash. Si la clave del elemento agregar $e.clave$, se encuentra en la tabla de hash, entonces se sustituye el valor del dato en la tabla de hash por el valor de $e.dato$.

agregar_elem(HashEntry e)

Agregar clave y dato: Se agrega a a la tabla de hash una clave c, que tiene asociada un dato de tipo String d . Si la clave a agregar se encuentra en la tabla de hash, entonces se sustituye el valor del dato en la tabla de hash por el valor de d.

agregar(int c, String d)

Eliminar elemento: Dada una referencia de un elemento e de tipo *HashEntry* de la tabla de hash, entonces se elimina a e de la tabla. Si e no es una referencia a un elemento en la tabla de hash, entonces la operación no tiene ningún efecto en la tabla.

eliminar_elem(HashEntry e)

Eliminar con clave: Dada un clave c, si algún elemento en la tabla de hash tiene una clave igual a c, entonces el elemento se elimina de la tabla y retorna el String asociado a esa clave. En caso de que no haya ninguna clave c en la tabla de hash, se retorna `None`.

eliminar(int c) $\rightarrow$ String

Buscar con clave: Dada un clave c, se busca el elemento en la tabla de hash que posea la clave igual a c. Si el elemento se encuentra en la tabla, entonces se retorna el String asociado a esa clave. En caso de que no haya ninguna clave c en la tabla de hash, se retorna `None`.

buscar(int c) $\rightarrow$ String

Mostrar el contenido: Se muestra por la salida estándar todos los elementos de la tabla de hash, en forma de pares clave y valor asociado.

MOSTRAR(void)

El método de hashing que usted debe usar, es el método de la división que se explicó en el curso de teoría de Algoritmos y Estructuras 2 [2], usando el valor que recomienda D. E. Knuth. Además de la Tabla de Hash, debe hacer un cliente que muestre el correcto funcionamiento de las operaciones que haya implementado. Son cuatro los archivos que debe entregar:

`hashEntry.py`**:** Contiene una clase con la implementación del tipo de datos *HashEntry*.

`dlist.py`**:** En este módulo debe estar implementado la lista doblemente enlazada con la clase *Dlist*. La lista debe contener elementos de tipo *HashEntry*.

`hash_table.py`**:** En este archivo se implementa el tipo de datos Tabla de Hash, haciendo uso de los tipo de datos *HashEntry* y *Dlist*.

`test_htable.py`**:** Cliente con el que se prueba y muestra el correcto funcionamiento de las operaciones de la Tabla de Hash.

2. Condiciones de entrega

La versión final del código del laboratorio debe estar contenidos en un archivo comprimido, con formato *tar.xz*, llamado *LabSem9_X.tar.xz*, donde X es el número de carné del estudiante. La entrega del archivo *LabSem9_X.tar.xz*, debe hacerse al profesor del laboratorio por email, antes de las 9:00 pm del día domingo 08 de marzo de 2020.

Referencias

[1] Cormen, T., Leiserson, C., Rivest, R., and Stein, C. *Introduction to algorithms*, 3rd ed. MIT press, 2009.

[2] Palma, G. Ci2612: Algoritmos y estructuras de datos ii. `https://gpalma.github.io/courses/ci2612em20/`, 2020.

Guillermo Palma / gvpalma en usb.ve / Febrero 2020