Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI2692 – Laboratorio de Algoritmos y Estructuras II
Enero–Marzo 2017

Tablas de Hash

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 y con la misma resolver un problema. 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 de igual manera. El encadenamiento debe ser llevado a cabo mediante una lista doblemente enlazada. La figura 1 muestra un ejemplo de una Tabla de Hash donde la clave es de tipo entero, la cual es semejante a la que se presenta en el Cormen et al [1].

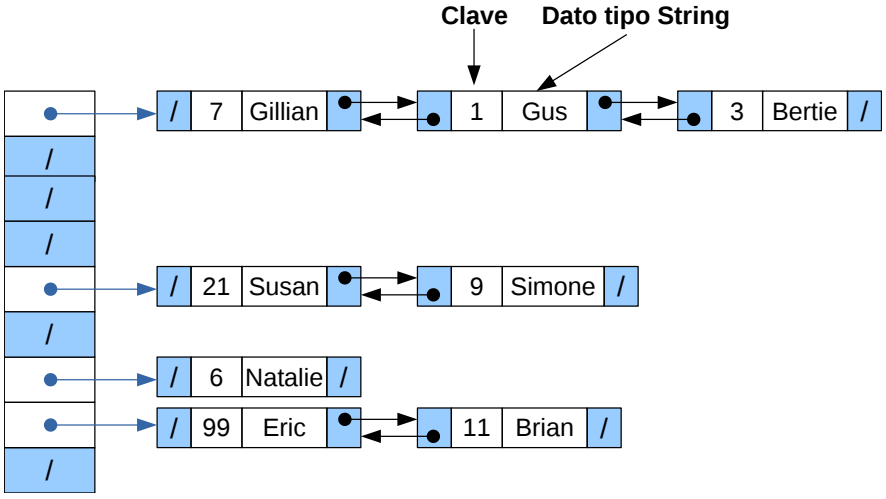

Figura 1: Ejemplo de la Tabla de Hash donde el encadenamiento es realizado con una lista doblemente enlazada. Las claves son elementos de tipo entero, y el dato asociado a cada clave es de tipo String.

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)

La clave que se le asignará a un dato vendrá dada por la función de Python *hash(str)*.

2. Problema

Recientemente algunos antisociales han intentado irrumpir en la base de datos de DACE para borrar todas las notas de los estudiantes con la finalidad de que todos tomen las materias nuevamente desde Matemáticas I, Lenguaje I, Venezuela ante el Siglo XXI-I e Inglés I (no importa si la haya eximido o no, deberá verla). Es por ello que es necesario establecer nuevas contraseñas a las máquinas para una máxima seguridad. Los estudiantes del Laboratorio de Algoritmos II quieren escribir un programa que chequee si la nueva contraseña que están estableciendo para entrar a las computadoras de DACE es segura.

Para que una contraseña sea segura deberá seguir las siguientes condiciones:

- Tiene que tener al menos 8 caracteres de largo
- Ninguna palabra podrá encontrarse en el diccionario
- No es una palabra del diccionario seguida por un dígito (hola1)
- No son dos palabras del diccionario separadas por un dígito (hola1javieres)

Las dós últimas condiciones son opcionales para realizar por el estudiante. Es decir, no es obligatorio implementarlas ya que no poseerán ningún puntaje en este laboratorio.

El diccionario de palabras es un archivo que tendrá el siguiente formato:

```
n
palabra_1
palabra_2
palabra_3
...
palabra_n
```

donde en la primera línea n es el número de palabras del diccionario, para luego en las siguientes n líneas se presenten las palabras del diccionario.

El programa deberá pedirle al usuario que introduzca la nueva contraseña que desea establecer. Previamente deberá cargar un archivo llamado diccionario.txt que poseerá las palabras del diccionario antes mencionado. La salida deberá ser por pantalla y deberá señalar si el usuario está colocando una contraseña correcta o no. En caso de no ser correcta deberá pedirle nuevamente al usuario que introduzca una nueva contraseña.

3. Archivos a entregar

Además de la Tabla de Hash y el problema resuelto, debe hacer un cliente que muestre el correcto funcionamiento de las operaciones que haya implementado. Son cinco 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*. En el libro [2] podrá encontrar una implementación de una lista doblemente enlazada.

`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.

`problema.py`: Archivo que contiene la solución al problema planteado.

4. Condiciones de Entrega

El trabajo es por equipos de laboratorio. Debe entregar los códigos fuentes de sus programas, en un archivo comprimido llamado LabSemana9-X-Y.tar.gz donde X y Y son los números de carné de los integrantes del grupo. La entrega se realizará por medio del aula virtual antes de la 11:59 pm del sábado 11 de marzo de 2017.

Referencias

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

[2] Necaise, R. *Data Structures and Algorithms Using Python*. Wiley, Inc., 2009.

Catherine Lollett, Guillermo Palma / catherinelollett@usb.ve, gvpalma@usb.ve / Enero 2017