Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI-2692 - Laboratorio de Algoritmos y Estructuras II
Trimestre Septiembre-Diciembre 2017

Implementación de una Tabla de Hash usando el Cuco Hashing

1. Introducción

El objetivo del laboratorio es realizar la implementación de la estructura de datos Tabla de Hash que usa el método de Cuco hashing para la resolución de colisiones.

2. Descripción de la tabla de Hash

Usted debe implementar una tabla que usa Cuco hashing en donde las claves son de tipo *String*, y el valor que esta asociado a cada clave, es un elemento de tipo *String*. La Tabla de Hash estática. Esto es, la tabla se crea con un tamaño determinado y este no cambia durante su vida de ejecución. La figura 1 muestra un ejemplo de la Tabla.

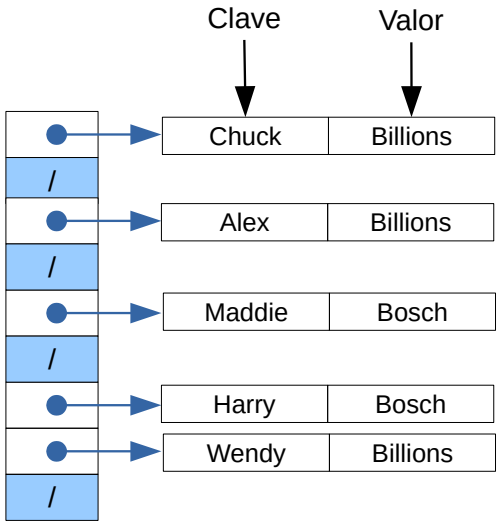

Figura 1: Ejemplo de la Tabla de Hash usando Cuco hashing. Las claves son elementos de tipo *String*, y el valor asociado a cada clave es de tipo *String*.

2.1. Clase *CucoEntry*

El tipo *CucoEntry* debe poseer dos campos: el primero se llama *clave* y el segundo se llama *valor* y ambos son del tipo *String*. El tipo *CucoEntry* debe ser implementado como una clase de Python, en el archivo `CucoEntry.py`. El constructor de la clase *CucoEntry* recibe como entrada un clave, de tipo *String* y un valor asociado con la clave, que tambien es de tipo *String*. A continuación la firma del constructor.

crearCucoEntry(String *clave*, String *valor*)

2.2. Clase *CucoTable*

La Tabla de Hash usando Cuco hashing debe ser implementada como una clase pública de Python, llamada *CucoTable*. El desarrollo de tabla debe seguir el pseudocódigo presentado en la clase del curso de teoría de Algoritmos y Estructuras 2 [1]. La tabla almacenará elementos de tipo *CucoEntry*. El arreglo que contiene a los elementos de tipo *CucoEntry* debe ser de tipo *arrayT*. Las operaciones de la tabla se hash son:

Crear tabla: Constructor de la clase, que crea una tabla con un tamaño inicial de n.

crearCucoTabla(int n)

Agregar clave y valor: Se agrega ala tabla de hash una clave c, que tiene asociada un valor v. 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 v. Usted debe prevenir que este método quede en un ciclo infinito, utilizando la condición adecuada para terminar el mismo. Si un elemento ha sido agregado de forma exitosa, se retorna el valor de 1, y en caso de no poder agregarse un elemento, entonces el método retorna None.

agregar(String c, String v)

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(String c) → 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(String c) → 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)

2.3. Funciones de hash

El método de Cuco hashing hace uso de dos funciones de hash h_1 y h_2. Para la función h_1 se empleará la función de hash SHA256 [1] . Para la función h_2 se utilizará el algoritmo MD5 [2]. En específico se debe hacer uso de las implementaciones de SHA256 y MD5 que vienen en en el módulo de Python3 `hashlib`. Para este laboratorio se les dará la implementación de las funciones h_1 y h_2. El listado 1 muestra el código de h_1 y h_2.

```
1 import hashlib
2
3 def h1(clave):
4     h = hashlib.sha256(clave.encode())
5     return int(h.hexdigest(),base=16)
6
7 def h2(clave):
8     h = hashlib.md5(clave.encode())
9     return int(h.hexdigest(),base=16)
```

Listado 1: Funciones de hash h1 y h2.

3. Actividad a realizar

Usted debe implementar en `Python3` la Tabla de Hash que usa el método de Cuco hashing, junto con las operaciones indicadas en la sección anterior. Además de la Tabla de Hash, debe hacer un cliente que muestre el correcto funcionamiento de las operaciones que haya implementado. Los archivos que debe entregar son los siguientes:

`arraT.py`: Módulo con la implementación del tipo de datos *ArrayT*.

`cuco_entry.py`: Contiene la implementación de la clase *CucoEntry*.

`cuco_table.py`: En este archivo contiene la implementación de la tabla de hash por medio de la clase *CucoTable*.

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

4. Condiciones de entrega

Debe entregar el día 16 de Marzo de 2017 antes de las 4:30 pm, un archivo comprimido llamado **LabSem10-A-B.tar.gz**, con todos los códigos en `Python3`. Las letras **A** y **B** del archivo comprimido son los número de carné de los integrantes del equipo.

Referencias

[1] Bonet, B. Ci2612: Algoritmos y estructuras de datos ii (pregrado). `http://maracaibo.ldc.usb.ve/~blai/courses/ci2612/`, 2017.

Guillermo Palma / gvpalma@usb.ve / Marzo 2017

[1] `https://en.wikipedia.org/wiki/Secure_Hash_Algorithm`
[2] `https://en.wikipedia.org/wiki/MD5`