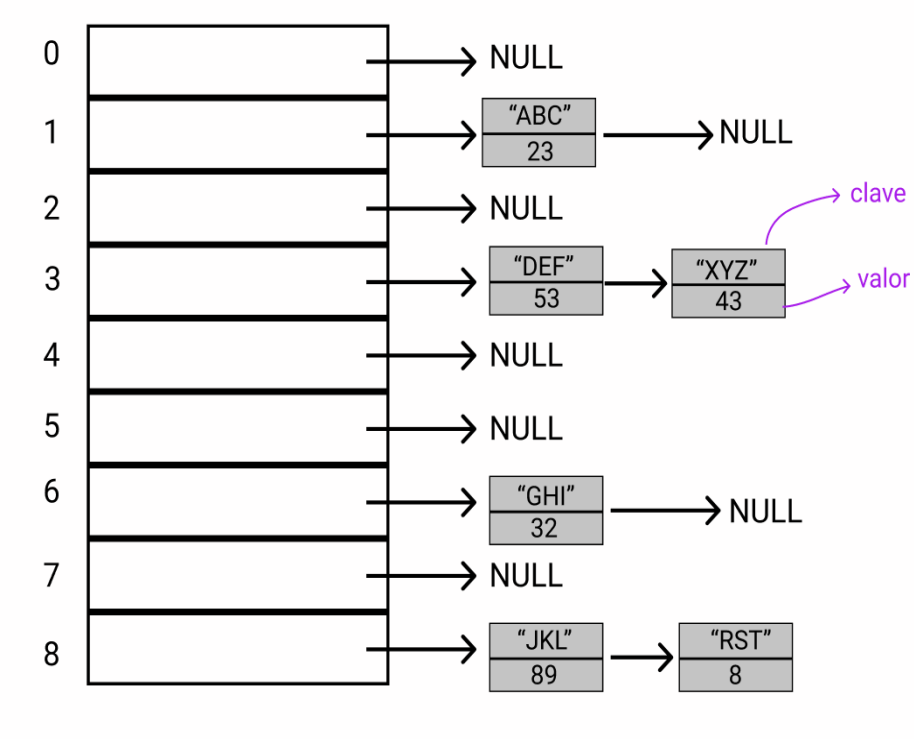
FACULTAD DE INGENIERIA
Universidad de Buenos Aires

TDA HASH

[7541/9515] Algoritmos y Programación II



Representación grafica de la tabla de hash abierta con direccionamiento cerrado.

Segundo cuatrimestre de 2021

Alumna: CONVERSO, Lara Daniela

Número de padrón: 107632

Email: lconverso@fi.uba.ar

1. Detalles de implementación

Para la implementación de este TDA, dado que se trata de un hash abierto con direccionamiento cerrado, decidí utilizar el TDA LISTA realizado con anterioridad en el transcurso de este cuatrimestre, ya que su implementación se complementa para obtener el TDA HASH esperado.
Para realizar el encadenamiento típico del hash abierto utilice listas enlazadas.

Hash_crear: Se encarga de crear el hash, desde reservar su memoria, inicializar sus variables y crear las listas que se encargaran de resolver el direccionamiento.

Hash_ insertar: Primero chequea si el factor de carga de la tabla es mayor al 0.75% de la tabla, si es mayor hace un rehash, que en mi implementación básicamente es generar una tabla con el doble del tamaño que la anterior y copiar todos los elementos.
Se encarga de crear un elemento con la clave y el valor que recibe pasado por parámetro. Luego recorre la tabla evaluando si la clave ya esta dentro, en el caso de estarlo, reemplaza el valor y mantiene la misma clave. En caso de no estar, inserta el elemento creado a la tabla (y respectiva lista).
Si el elemento fue insertado con éxito, se aumenta la cantidad de elementos en la tabla por uno.

Hash_quitar: Se encarga de retirar un elemento de la tabla dada una clave. Para ello busca la clave dentro de la tabla y una vez encontrada libera la memoria reservada para esta y su elemento y le resta a la cantidad de la tabla un elemento.

Hash_contiene y Hash_obtener: Ambas funciones, operan de manera similar, recorren la tabla de hash, hasta encontrar la clave deseada, en el caso de contiene, el resultado es true o false si se encuentra o no la clave; Para obtener el resultado de la función es el elemento asociado a la clave que estaba siendo buscada.

Hash_cantidad: retorna la cantidad de elementos almacenados en la tabla de hash.

Hash_destruir: se encarga de liberar la memoria reservada para el hash incluyendo los elementos dentro.

Hash_con_cada_clave: Se encarga de recorrer los elementos insertados en el hash e ir contando los elementos recorridos, su recorrido depende de una función pasada por parámetro si esta devuelve true, el recorrido se corta.
Para esta función utilice las funciones de lista, para obtener la cantidad de elementos a recorrer y para evaluar cada elemento con la función pasada por parámetro.

2. Compilación

Para la compilación se utilizo el `makefile` de la cátedra, los comandos correspondientes, para correr el archivo con las pruebas propias y valgrind se utiliza: `'make' o 'make valgrind-pruebas',` y para compilar con el archivo de ejemplo que brinda la cátedra se utiliza `'make valgrind-ejemplo'.`

3. Pruebas

En el archivo de pruebas.c se encuentran las pruebas que se utilizaron para la implementación de este TDA, se utilizo la metodología TDD, donde primero creo las pruebas dependiendo del resultado que se quiere y luego se implementan las funciones.

4. Preguntas teóricas

Definir tabla de hash. Enumerar los distintos tipos de tablas que conoce y explique cómo se diferencian en cuanto a funcionamiento de las diferentes operaciones.

La tabla de hash es una estructura que se encarga de guardar elementos, estos están asociados con una clave para poder encontrarlos.
Su función es ir acomodando los elementos con sus respectivas claves y valores dentro de una tabla, para ello utiliza una función (que varia dependiendo la implementación), que se encarga de ubicar dentro de la tabla el elemento a insertar. Básicamente toma la clave del elemento a insertar y la transforma en un numero que esté asociado a las posiciones disponibles según la capacidad de la tabla.
Si tengo dos claves iguales, el valor de este elemento debe ser actualizado. En el caso de tener dos claves distintas que dan igual resultado en la función de hash, se produce una colisión.
Si bien puede parecer un problema, existen dos formas de re-direccionar el elemento que origina esta colisión. Estas son el direccionamiento cerrado, y el direccionamiento abierto, el tipo de hash que tenemos dependen de estos direccionamientos.

- Direccionamiento abierto, clasifica al hash como cerrado.
- Direccionamiento cerrado, clasifica al hash como abierto.

El hash abierto, como el de mi implementación, utiliza un encadenamiento, el cual implemente con una lista enlazada. Cada nodo/elemento esta conectado al índice de la tabla que le corresponde.
El hash cerrado, con direccionamiento abierto como todos los valores se guardan dentro de la misma tabla, si se genera una colisión se resuelve recorriendo la tabla hasta que se encuentre el siguiente espacio disponible para poder guardar el elemento.
Otro aspecto a tener en cuanta es el factor de carga de la tabla del hash, cuanta mas capacidad posea la tabla menos probabilidades hay de que ocurra una colisión.