.UBA fiuba
FACULTAD DE INGENIERÍA

ALGORITMOS Y PROGRMACIÓN II, CURSO 04-MENDEZ
INFORME DEL TDA#3: HASH.

Alumno: Víctor Zacarías, Padrón: 107080

DNI: 95.979.449, Mail: vzacarias@fi.uba.ar

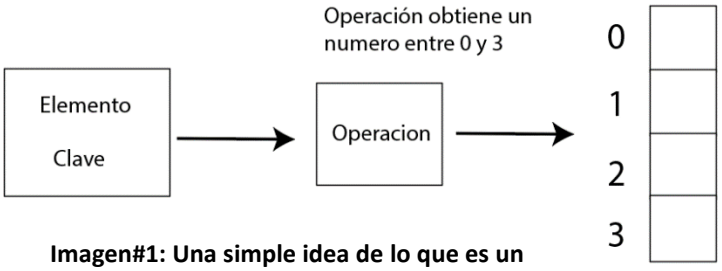
-Descripción: En este informe se incluye una breve explicación teórica sobre el tda hash, así como explicaciones sobre la implementación de este (se omitieron explicaciones sobre funciones cuyo funcionamiento es sencillo).

-introducción teórica:

Un hash o tabla de hash es un tipo de dato abstracto que permite almacenar elementos en una estructura similar a un arreglo. Si bien una tabla de hash tiene cierta semejanza con un arreglo, tiene una serie de características que la definen y distinguen de los arreglos:

- Cada elemento de un hash es un par elemento-clave.
- Al momento de implementar un hash se debe implementar una operación que se realice sobre las claves.
- Esta operación determina en que posición de la tabla se encuentran o deberán ser insertados los elementos. De esta forma a cada elemento le corresponderá una posición en la tabla de hash determinada por la clave a la que se encuentra asociado.

¿Qué ventajas trae trabajar con un hash en lugar de trabajar con un arreglo?: El tener una operación que se aplique sobre cada calve de los elementos permite determinar con rapidez en que posición de esta se encuentran o deberán ser insertados, esto es, disminuye el tiempo de búsqueda e inserción de elementos.



Imagen#1: Una simple idea de lo que es un hash.

En imagen#1 se puede visualizar a la derecha la tabla de hash, a la izquierda un elemento (asociado a una clave) que se quiere insertar, la operación sirve para determinar en qué posición de la tabla deberá ser insertado el elemento. Nótese que si se quiere buscar el elemento insertado la operación nos dice en que posición se encuentra y no hay necesidad de recorrer linealmente la tabla (como en el caso de un vector).

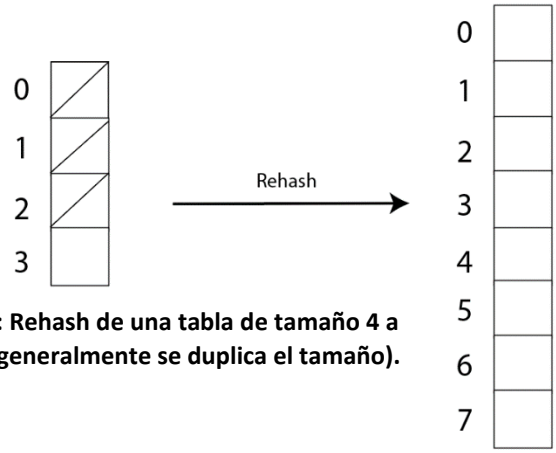
- **función hash:** Es una función (en la imagen#1, operación) que a partir de una clave (que debe estar relacionada con un elemento) determina qué lugar de la tabla le corresponde a un elemento.

En una tabla de hash se insertan elementos en la tabla, lo que genera un problema al momento de determinar el tamaño de esta. Es por esto que se define el factor de carga y el rehash, que permiten aumentar el tamaño de la tabla en el momento adecuado.

- **Factor de carga (α):** Es un numero entre 0-1 que determina que tan llena esta la tabla de hash.

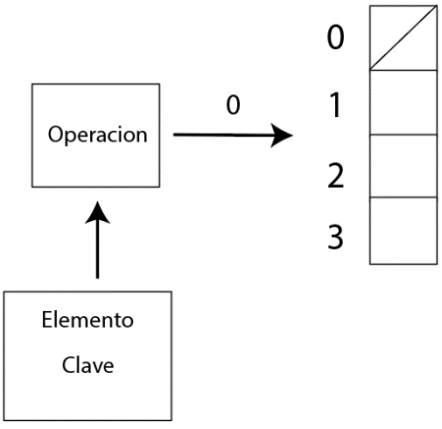
  Factor de carga = cantidad de elementos / capacidad de la tabla

- **Rehash:** Es una operación que se realiza cuando el factor de carga es muy alto (generalmente cuando es mayor o igual que 0,75), en esta básicamente se crea una nueva tabla de hash cuyo tamaño es el doble (generalmente) que el anterior, y los elementos de la tabla vieja se insertan en la nueva. Para esta nueva tabla de hash la función hash debe poder determinar en qué posición deben ser reinsertados los elementos.



Imagen#2: Rehash de una tabla de tamaño 4 a una de 8 (generalmente se duplica el tamaño).

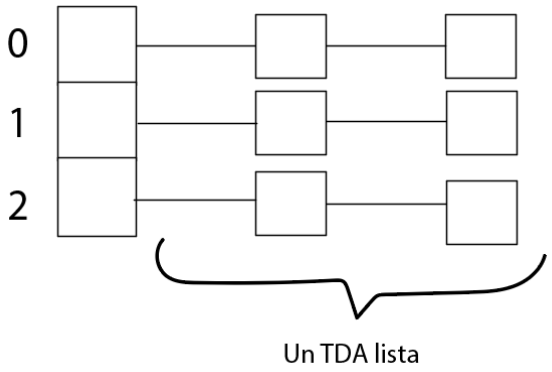
- **Colisiones:** En un hash cuando se insertan muchos elementos, existe la posibilidad de que para una clave se le asigne una posición ya ocupada en la tabla, el cómo proceder con las colisiones depende del tipo de hash con el que se trabaje.



Imagen#3: Quiero insertar un elemento en una posición que ya está ocupada.

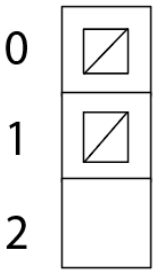
- **Tipos de hash:**

    - **Hash abierto (de direccionamiento cerrado):** En un hash abierto los elementos a insertar se guardan por fuera de la tabla (para poder hacer esto se usan como auxiliares alguna estructura o tipo de dato abstracto). Se le dice de direccionamiento cerrado porque en caso de una colisión inserta un elemento al lado del otro (ver imagen#4).

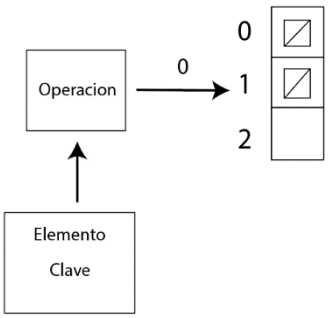


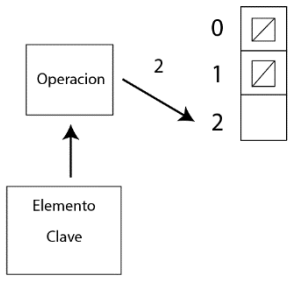
Imagen#4: Un hash abierto, los elementos se guardan afuera de la tabla usando una lista. Se puede apreciar que es de direccionamiento cerrado (los elementos colisionados se insertan uno al lado del otro).

    - **Hash cerrado (de direccionamiento abierto):** En un hash cerrado los elementos se guardan dentro de la tabla de hash. Se le dice de direccionamiento abierto porque en caso de una colisión su usa un criterio para determinar en qué posición de la tabla debe ser insertado el elemento (en general se utiliza el probing lineal, en donde se inserta el elemento en la próxima posición libre, el caso en que no haya posiciones libres no se da gracias al factor de carga y rehash).



Imagen#5: Un hash cerrado, se logra apreciar que los elementos de guardan dentro de la tabla.





Imagen#6: El direccionamiento cerrado del hash abierto. En la imagen se usa probing lineal para solucionar el problema.

La diferenciación en un tipo de hash y otro se centra en cómo resolver las colisiones, el resto de las operaciones que se hacen en un hash son similares. Estas operaciones se detallan en la explicación de la implementación realizada para el TDA.

-**Explicación de la implementación realizada:**

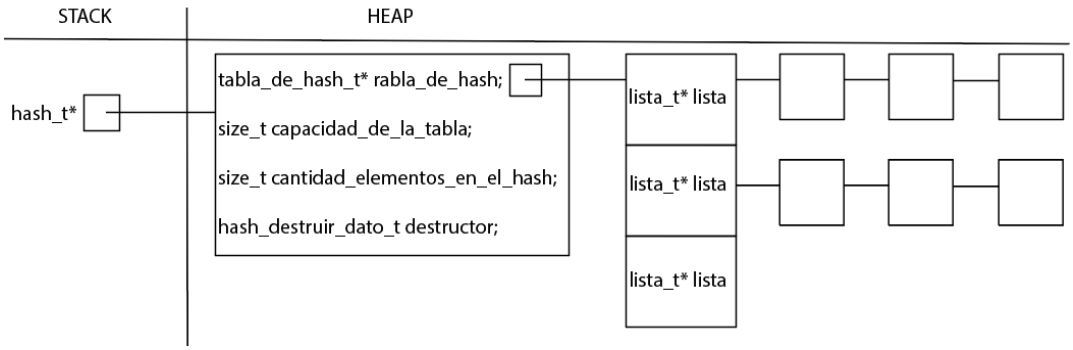
En esta implementación del TDA hash se hizo uso de un hash abierto, para lo cual se hizo uso de una lista para almacenar lo elementos fuera de cada casillero de la tabla. La idea de usar una lista para almacenar los elementos se centra en que es fácil trabajar con ella al momento de manejar las colisiones dentro del hash (en este caso el elemento se inserta en la próxima posición libre de la lista), por otra parte, el hash, al estar regulado por el factor de carga impide que alguna lista llegue a tener una gran cantidad de elementos, por lo que el tiempo de búsqueda es corto.

Al usar el TDA lista, se aprovechan también cada una de las operaciones (crear lista, insertar, etc.) que la lista provee.

NOTA#1: En buena parte de explicación se referirá a los elementos insertados en el hash como objetos, para distinguirlos de elemento; En un hash el usuario inserta un elemento asociado a una clave, un objeto es un registro que contiene dos campos, el elemento y la clave. Los objetos son insertados en el hash.

NOTA#2: Al aprovechar las funcionalidades de la lista se hizo especial uso de los iteradores aportados por la misma. El usar los iteradores permite recorrer la lista una única vez al realizar algunas operaciones, como de búsqueda, obtención, etc.

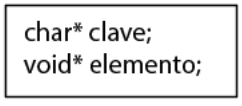
Por ejemplo, si se quiere buscar un elemento con la clave “HOLA” y se usa la función lista_elemento_en_posicion se recorrerá potencialmente x veces toda la lista hasta encontrar un elemento con clave “HOLA. El uso de iteradores permite desplazarse por la lista y ver sus elementos posición a posición.



Imagen#7: Estructura usada para implementar el TDA hash.

Destructor es una función aportada por el usuario que se le aplica a cada elemento (no la clave) insertado en el hash (puede ser NULL en caso de que el usuario lo desee).

En la imagen puede apreciarse como cada posición de la tabla contiene una lista.



Imagen#8: Estructura de tipo elemento_del_hash_t* de los elementos que son insertados en el hash. Cada elemento se encuentra asociado una clave.

-Funciones auxiliares: Funciones usadas para modularizar las funciones públicas del TDA hash.

1-Funcion factor_de_carga: Determina el factor de carga del hash, para esta implementación se calcula como:

cantidad_de_elementos_en_el_hash/capacidad_de_la_tabla

Se toma en cuenta la cantidad total de elementos en el hash y no la cantidad de casilleros ocupados de la tabla para determinar este factor.

2-Funcion funcion_hash: Su objetivo es el de determinar que posición de la tabla le corresponde a una clave. Debe recibir la clave (debe ser un string) y la capacidad de la tabla.

Para su implementación se recorre cada letra del string (sin contar el \0) y se suman su valor en ASCII, a esa cantidad total se le calcula el resto con la capacidad de la tabla. De esta forma se obtiene un numero entre 0 y capacidad_de_tabla-1 que representa que posición de la tabla le corresponde a la clave dada.

3-Funcion destruir_elementos_de_lista: La función se encarga de destruir cada elemento_del_hash_t* (llámese objetos) insertado en la lista de un casillero de la tabla de hash, para esto hace uso de los iteradores de lista, que permiten recorrer elemento a elemento una lista mientras se libera la memoria usada por los objetos insertados en ella (la lista se recorre una única vez). La función no destruye la lista en sí, sino los objetos que están en ella.

4-Funcion hash_insertar_aux: Es una función auxiliar que se encarga de insertar elementos dentro de un hash.

- Recibe el hash en el que se quiere insertar, la clave, y el elemento a insertar.
- Dada la clave y a partir de funcion_hash determina en que posición de la tabla de hash deberá ser insertado el nuevo elemento.
- Crea un registro del tipo elemento_del_hash_t* donde guarda elemento y una copia de clave (así el usuario no puede modificar las claves insertadas).
- Verifica si en la posición en la que se quiere insertar existe una lista, en caso de que no exista se crea.
- Finalmente, el elemento es insertado en la posición correspondiente usando lista_insertar.

hash_insertar_aux se encarga de manejar las colisiones a través de la función lista_insertar.

5-Funcion posicion_de_elemento: Se encarga de verificar si una clave se encuentra en una lista de una posición de la tabla de hash. Para esto hace uso de los iteradores de lista, que permiten recorre elemento a elemento una lista verificando si la clave se encuentra en ella. Devuelve la posición de la lista en la que se encuentra el elemento asociado a la clave, en otro caso devuelve -1.

6-Funcion iterar_hasta: Recibe una lista de una posición de la tabla de hash y una clave. Se encarga de recorrer la lista usando iteradores de lista hasta encontrar el elemento de la lista que contenga la clave buscada. En ese caso se devuelve el objeto de lista que contiene la clave (elemento_del_hash_t*), en caso de no encontrar la clave devuelve NULL.

La utilidad de esta función es la de obtener un elemento de la lista conociendo su clave.

7-Funcion liberar_tabla_de_hash: Recibe una tabla_de_hash y su capacidad. La función se encarga de recorrer cada posición de la tabla_de_hash liberando la memoria ocupada por ella, hace uso de destruir_elementos_de_lista para liberar lo elementos del hash, y de lista_destruir para liberar la memoria ocupada por cada lista de la tabla.

Cada vez que se itera sobre una posición de la tabla, se liberan los elementos contenidos en ella, y posteriormente se libera la lista (si es que estos existen, en caso contrario no pasa nada). Este proceso se repite hasta recorrer toda la tabla.

8-Funcion reinsertar_elementos_de_hash: Su objetivo es el de insertar los elementos de un casillero de una tabla de hash en otra tabla de hash, para esto recibe una lista con los elementos que se quieren insertar en otra tabla y el hash en el que quieren insertar. La función hace uso de iteradores de lista para recorrer elemento a elemento la lista, obteniendo cada elemento de ella e insertándolo en el nuevo hash (haciendo uso de hash_insertar_aux).

Esta función es de gran utilidad para el funcionamiento del rehash (ver función rehash).

9-Funcion rehash: En caso de que el factor de carga exceda el valor establecido (en este caso 0,75) se deberá hacer un rehash.

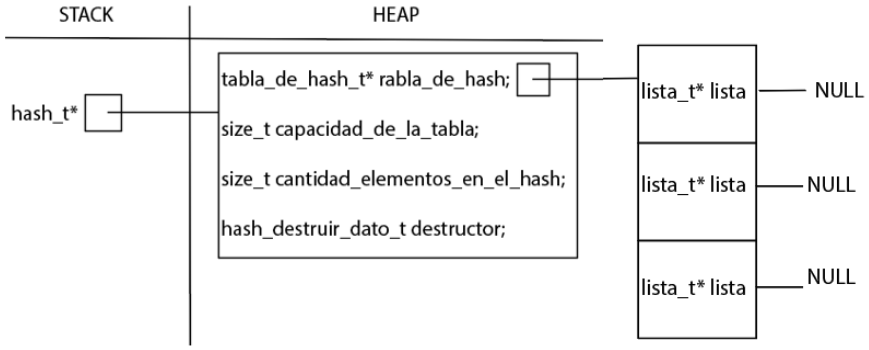
La función recibe el hash a rehashear. Para cumplir con su objetivo, la función crea un hash_auxiliar con el doble de la capacidad que el hash a rehashear, luego, recorre iterativamente cada posición de la tabla de hash del hash a rehashear, usando en cada iteración la función reinsertar_elementos_de_hash para insertar los elementos de cada lista de la tabla en el hash_auxiliar.

Finalmente, se libera la tabla de hash del hash rehashear usando la función liberar_tabla_de_hash, y se actualiza el hash a rehashear con la información de hash_auxiliar.

El hash_auxiliar permite reinsertar todos los elementos de un hash en un nuevo hash cuyo tamaño es el doble que el anterior.

-Funciones públicas: Se omiten aquellas funciones cuyo funcionamiento es simple y se resumen en pocas líneas.

1-Función hash_crear: Se encarga de crear un hash para poder ser usado por el usuario, este hash debe ser creado por un tamaño especificado por el usuario (si el tamaño es menor que 3, el hash por defecto será creado con tamaño 3, con tamaño se hace referencia a la capacidad de la tabla).



Cuando un hash es creado cantidad_elementos_en_el_hash es 0. Destructor es una función que del usuario que es aplicada a cada elemento insertado en el hash (si el usuario no desea usar un destructor, debe mandar NULL en lugar de una función).

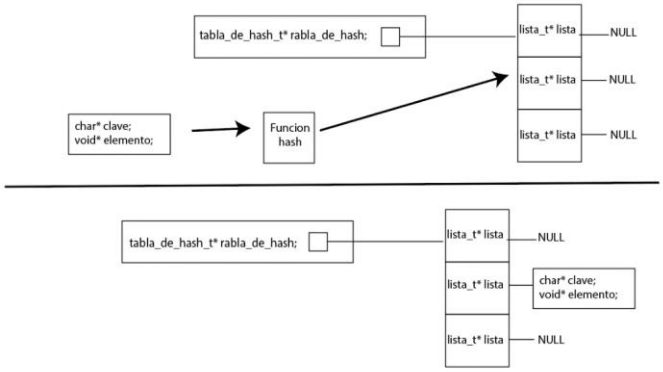
2-Funcion hash_insertar: Dado un hash una clave y un elemento, se encarga de insertar un registro del tipo elemento_del_hash_t* (llámese objeto) que contenga a clave y elemento dentro del hash. Esta función tiene 3 caso importantes:

- Inserción simple: Se inserta un elemento en una posición de la tabla de hash, en caso de haber una colisión este es insertado al lado del elemento colisionado (para esto se usa una lista).
- Clave repetida, si se quiere insertar un elemento que este asociado a una clave que se encuentra en el hash, la función actualiza el elemento que se encuentra dentro del hash (en un registro del tipo elemento del hash_t*) con el elemento a insertar.
- Rehash: Cuando al hacer una inserción, el hash excede el factor de carga.

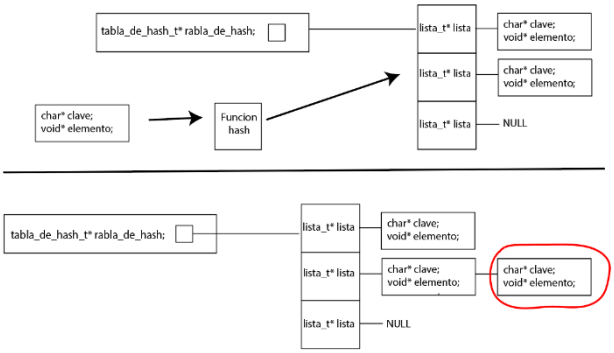
*Una breve descripción de la función:

1. La función se encarga de verificar si la clave asociada a elemento ya se encuentra en un objeto en el hash.
    a. En caso afirmativo, actualiza el elemento que se encuentra en el hash con el que se quiere insertar. Le aplica el destructor (si no es NULL) al elemento viejo. Para esto se usa la función iterar hasta.
2. En caso de que la clave no se encuentre ya en el hash, se usa la función hash_insertar_aux para realizar la inserción.
3. Luego de insertar el elemento, se verifica cual es el factor de carga del hash, y en caso de que sea mayor o igual que 0,75, se usa la función rehash para rehashear el hash.

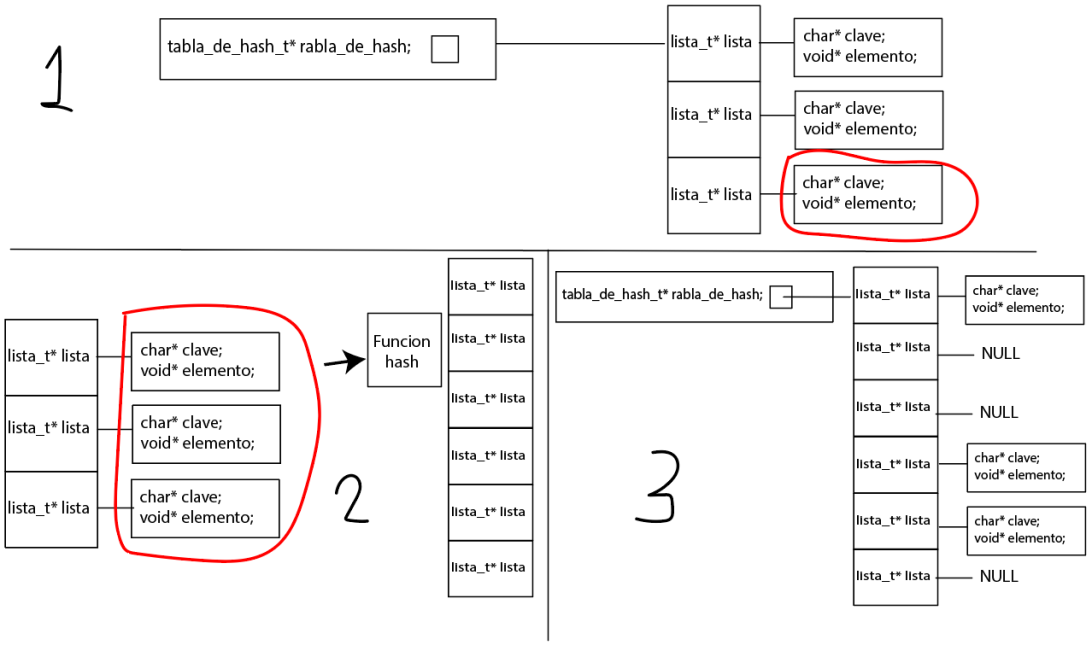
NOTA: 2 y 3 solo se ejecutan si 1 no fue ejecutado.



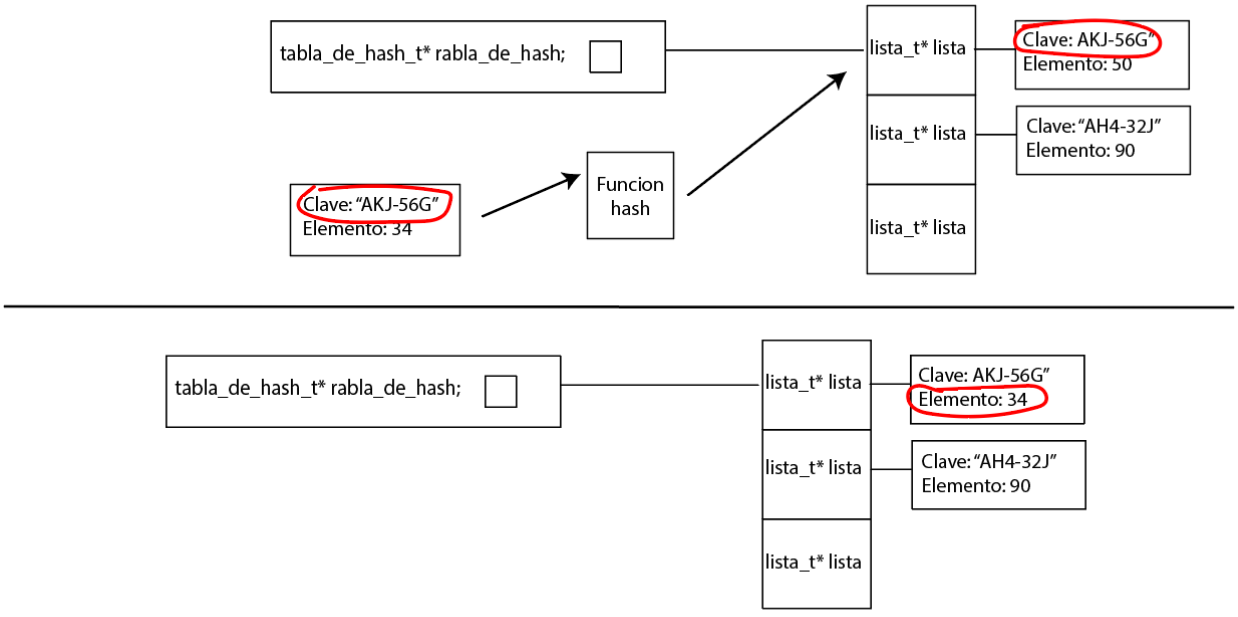
Imagen#9: Inserción simple, se quiere insertar un elemento, la función hash determina en que posición de la tabla se deberá insertar la estructura con la clave. El elemento se inserta usando la función lista_insertar.



Imagen#10: Inserción con colisión.



Imagen#11: Rehash. En este caso la tabla está al 100% de su capacidad, por lo que se rehashea, para ello, se crea una nueva tabla de hash (estructura del tipo de dato tabla_de_hash_t*) en donde se reinsertan los objetos de la tabla vieja. Finalmente, la tabla vieja es eliminada y reemplazada por la nueva. Para



Imagen#12: Inserción con clave repetida. En la imagen puede apreciarse como se intenta insertar un objeto con una clave repetida, en este caso, solo se actualiza el elemento del objeto contenido en el hash.

3-Funcion hash_quitar: Se encarga de quitar un objeto de la tabla, para esto recibe una clave asociada al objeto a quitar. Usa la función funcion_hash para determinar en que lugar de la tabla se encontrara el objeto, luego usa la función posicion_de_elemento para determinar en que lugar de una lista (de un casillero de la tabla) se encuentra el elemento. Finalmente usa la función lista_quitar_de_posicion para eliminar el objeto de la lista y luego libera la memoria ocupada por este.

4-Funcion hash_obetener: Dada una clave, obtiene un elemento dentro de un objeto asociado a dicha clave (registro del tipo elmento_del_hash_t*) insertado en el hash, para esto hace uso de la función funcion_hash para saber en que posición de la tabla buscar, y de la función _iterar_hasta para buscar el elemento en un casillero de la tabla de hash.

5-Funcion hash_contiene: Dada una clave y un hash se encarga de verificar si la clave se encuentra dentro del hash o no. Para esto hace uso de la función funcion_hash para verificar en que posición de la tabla buscar la clave, luego usa la función posicion_de_elemento para determinar si la clave se encuentra dentro del hash (esta función auxiliar retorna -1 en caso de que no se encuentre lo que se busca).

6-Funcion hash_destruir: Destruye el hash y toda la memoria ocupada por él. Para esto recorre la tabla usando destruir_elementos_de_lista (se liberan las claves, a los elementos se les aplica el destructor en caso de que este no sea NULL) y lista_destruir en cada posición. Luego se libera la tabla y la estructura que la contiene.

7-Funcion hash_con_cada_clave: Le aplica una función a cada clave del hash hasta que esta retorne true. La función devuelve el numero de veces que fue usada. Para poder hacer esto se recorre iterativamente la tabla de hash, en donde por cada posición se usan iteradores de lista para recorrer las listas de cada posición de la tabla, es acá donde se obtienen los objetos del hash y se les aplica la función.