TDA Hash

[7541/9515] Algoritmos y Programación II
Segundo cuatrimestre de 2021

Alumno:	Aldazabal, Lucas Rafael
Número de padrón:	107705
Email:	laldazabal@fi.uba.ar

Índice

1. Introducción

En este TDA se hace la implementación de una tabla de hash abierto con direccionamiento cerrado. Esta implementación cuenta con las siguientes operaciones:

- Crear.
- Insertar.
- Obtener.
- Quitar.
- Ejecutar una función a cada clave.
- Destruir eliminando los elementos.
- y operaciones auxiliares como obtener tamaño y ver si contiene una clave.

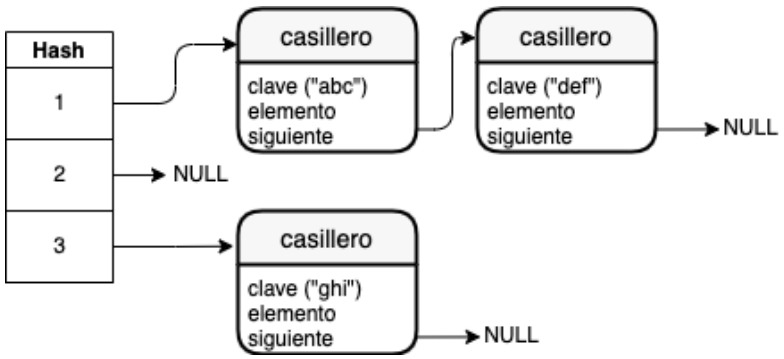
2. Teoría

2.1. Tabla de Hash

La tabla de hash es un tda que almacena datos asociados a una clave dada. Su principal beneficio es manejar de manera muy eficiente las búsquedas mediante la clave del dato buscado. Consiste en transformar la clave mediante una función llamada función hash en el numero que identifica su posicion el la estructura. Hay dos tipos de tablas de Hash, las de hash abierto (direccionamiento cerrado) o hash cerrado (direccionamiento abierto).

2.2. Hash Abierto

En este tipo de tablas los elementos no se almacenan dentro de la tabla como tal, esta es únicamente para los hashes y estos dirigen a otro tda donde se almacenan los datos y de ahí el concepto de hash abierto. En el caso de colisiones, es decir que dos claves distintas generan un mismo hash, ese tda almacena a los dos (por ejemplo una lista). Se les llama también direccionamiento cerrado porque el hash que nos da la funcion siempre es el hash de ese par clave-dato sin importar que colisione. Para buscar elementos que colisionaron lo que se hace es recorrer el tda de la posicion dada por la funcion hash hasta encontrar el elemento con la clave buscada.



2.3. Hash Cerrado

En el caso de los Hash cerrados, los datos y claves se almacenan también en la tabla, de ahí lo de cerrado, y en caso de colisión se busca la siguiente fila libre en la tabla haciendo que no siempre el índice de la funcion hash termine siendo la posicion real del par clave-dato. En estos casos a la hora

de buscar el elemento se comienza por la fila dada por la funcion hash y se busca para abajo hasta encontrar la fila con esa clave, esta búsqueda en cerrado se llama "probing lineal"pero también existen otros como "Hash doble"que consiste en tener una segunda funcion de hash para utilizar en estos casos o "probing cuadrático"que es buscar con la formula (intentos fallidos) elevado a 2

#	Clave	Elemento	funcion hash
1	abc	elemento	1
2	def	elemento	1
3	ghi	elemento	3

2.4. Función Hash

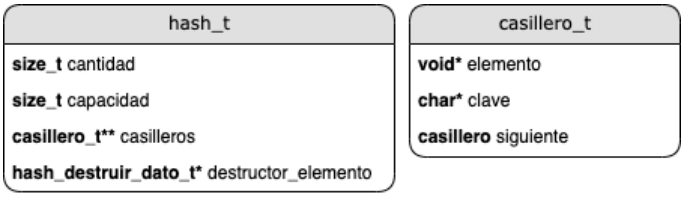
Como se menciono anteriormente, hay una funcion que se ocupa de definir la posicion en la tabla en base a la clave, estas pueden ser muy variadas y dependen de cada implementación, pero consiste en transformar los caracteres, números o lo que contenga la clave en un entero desde cero hasta el anterior al tamaño de la tabla de hash. Un ejemplo muy simple podría dividir una clave numérica por el tamaño de la tabla y usar el resto.

2.5. Función Rehash

En este tda lo mas importante es la eficiencia a la hora de buscar los elementos, siendo que en el ideal de que no haya ninguna colisión esta seria O(1), o sea buscar el índice en la funcion hash y ahí está. El peor caso seria tener todos los elementos colisionados haciendo que la complejidad se eleve a O(n), por esto es que lo que se hace es que cuando la cantidad de elementos crece hasta cierto porcentaje de la tabla y esto aumenta demasiado la posibilidad de colision es ampliar el tamaño de la tabla y recalcular las posiciones de los elementos existentes, esto va a permitir que se repartan mejor por la tabla y mejore mucho la eficiencia de búsqueda. Generalmente esto se hace cuando la cantidad de elementos alcanzan un valor >= a la capacidad ya que el rehash es muy costoso.

3. Detalles de implementación

Para esta implementación de crearon dos estructuras de datos, la tabla de hash y los casilleros.



3.1. Crear

Cuando se crea un Hash se recibe por parámetro la capacidad inicial y una funcion destructora de elementos para liberar estos de memoria cuando se eliminan de la tabla (puede ser NULL y que estos sigan en memoria luego de quitarlos), esta funcion va a recibir el elemento para que el usuario defina como debe liberarse. Luego se reserva en el Heap la memoria para el hash y se lo

inicializa con una tabla del tamaño recibido (si es menor a tres se inicia en tres) y con los valores en 0. Finalmente se devuelve el puntero del mismo.

3.2. Función Hash

La implementación de funcion hash utilizada consiste en utilizar los caracteres de la clave en su numero de la tabla ascii y sumar la multiplicación del primero por el ultimo con la del segundo y el ante ultimo y así recorriendo todos los caracteres y a esto dividirlo por la capacidad y utilizar el resto.

3.3. Insertar

Al insertar una clave con su elemento en la tabla lo primero que se hace es llamar a la funcion hash para obtener su índice y buscar esa posicion en el vector de casilleros (la tabla), si esa ubicación esta vacia, se crea un casillero y se almacena su puntero ahí, caso contrario se mira si el casillero en colision tiene un siguiente y se recorren los nodos (casilleros) hasta llegar al final y agregar el nuevo. Si mientras se recorren los casilleros anidados se encuentra uno con la clave recibida, lo que ocurre es que se utiliza el destructor del hash con el elemento anterior y se reemplaza por el recibido. Luego se revisa si la cantidad de elementos alcanza el 75 % de la capacidad y en ese caso se hace el rehash. Devuelve 0 si se pudo insertar y -1 en caso de error. Un dato a recalcar es que las claves se almacenan como copias así que el usuario no puede modificarlas por fuera del tda porque rompería el funcionamiento del mismo, si se quisiera cambiar alguna clave de un elemento se debe eliminar y reinsertar con la nueva.

3.4. Función Rehash

Para la funcion rehash lo que se hace es crear una nueva tabla con el doble de capacidad y luego recorrer la actual por cada lugar y los anidados insertando cada uno en la nueva de la misma manera que en insertar y liberando los casilleros viejos. esto es porque al cambiar el tamaño también hay que rehacer todos los hashes porque estos dependen de la capacidad, y no solo no podrías encontrar mas la clave porque el la búsqueda tendrías otro índice, sino que no solucionaría las colisiones anteriores.

3.5. Obtener

La implementación de la búsqueda es muy similar a la inserción, lo primero es calcular el índice de esa clave y buscar el casillero en esa posicion, si su clave es la buscada se devuelve su elemento y sino se revisa su siguiente hasta encontrar el de la clave o, si ninguno la tiene o el lugar estaba vacío directamente se devuelve NULL.

3.6. Quitar

A la hora de quitar, también se comienza calculando el índice y revisando el casillero en esa ubicación, si ese tiene la clave pedida se revisa que si tiene casillero siguiente ponerlo en la posicion inicial de la tabla para que no se pierda el acceso y luego se libera la clave, casillero y se utiliza el destructor en el elemento. Si el primer elemento no es el buscado se revisa los siguientes y si es algún de esos se redirige el siguiente del anterior al siguiente del que queremos quitar y se libera de la misma forma. En caso de que no se encuentre un elemento con esa clave o haya algún fallo se devuelve -1 y sino 0.

3.7. Recorrer claves ejecutando una función

Otra de las operaciones del hash es recorrer las claves almacenadas ejecutando una funcion dada por el usuario a la que se le pasa por parámetro el hash, la clave y una variable auxiliar a cada

una. Esta funcion tiene que devolver un booleano siendo false el esperado y true una indicación de cortar el recorrido. La funcion devuelve la cantidad de elementos recorridos. Este seria la manera de hacer un similar a un for o while para recorrer el hash ya que no hay manera de acceder por un índice ordenado (0, 1, 2, 3, ...) para el usuario.

3.8. Destruir Todo

La ultima operación necesaria de un hash es poder destruirlo para no perder memoria cuando se termina de ejecutar el programa. Para esto se recorre la tabla del hash eliminando los casilleros en cada posicion y sus siguientes en caso de haberlos, también se utiliza, en caso de que exista, la funcion destructora que se le indico al hash en su creación para liberar los elementos.

4. Otras operaciones implementadas

Se crearon dos funciones mas, una que devuelve el tamaño que tiene el hash, es decir la cantidad de claves/casilleros que tiene actualmente y otra que devuelve un bool si el hash contiene o no una clave, siendo esta una manera de utilizar la operación de búsqueda devolviendo true si se encuentra un elemento y false si no.