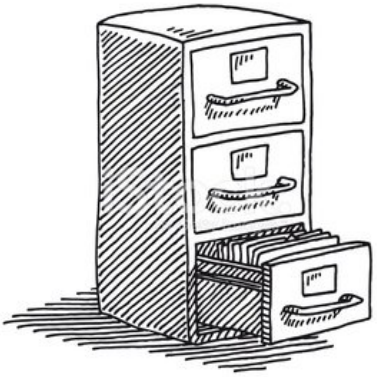
TDA Tabla de Hash



[7541/9515] Algoritmos y Programación II

Primer cuatrimestre de 2022

Autor:	**Email**:	**Padrón**:	**Fecha**:
Luca Lazcano	llazcano@fi.uba.ar	107044	19/06/2022

Índice

1. Introducción **2**

2. Tabla de hash **2**

2.1. Hash abierto 3

2.2. Hash cerrado 5

3. Implementación y decisiones de diseño **7**

1. Introducción

Para el último TDA visto en la cátedra, se tuvo que implementar una tabla de hash, con las especificaciones brindadas por la cátedra.

Además, se acompaña la implementación con este documento, donde se explora teoría pertinente al TDA tratado. Se explicará qué se entiende por tabla de hash abierta y cerrada; y sus principales operaciones.

Asimismo, se explicará la implementación realizada y decisiones de diseño tomadas.

Fueron provisto el archivo `hash.h` con las firmas y descripciones de las funciones a desarrollar, y un archivo `pruebas.c` donde se deben implementar pruebas para verificar el funcionamiento esperado de la implementación.

2. Tabla de hash

Una tabla de hash, también conocido como diccionario, es una estructura de datos que consiste de un mapeo desde un conjunto de claves únicas, a posiciones de un vector.

Para determinar la posición donde se guarda o busca un elemento, se ingresa la clave a una función de hash, que determinará la posición en el vector donde debería ir o estar el elemento. Para una clave y un tamaño del vector dado, el resultado del hasheo es único.

No sucede lo contrario: varias claves distintas puede ser hasheadas a la misma posición. Este suceso se llama colisión, y según cómo se resuelven, se puede separar a las tablas de hash en dos tipos: abiertas y cerradas.

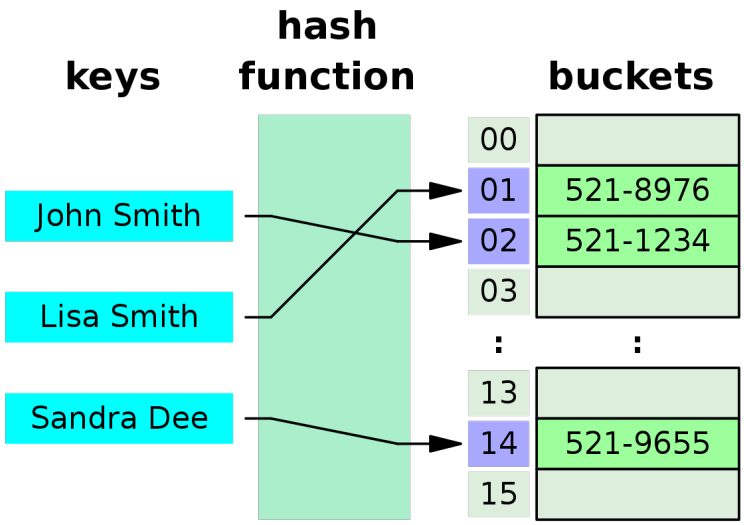


Figura 1: Tabla de hash.

2.1. Hash abierto

Cuando una clave es hasheada a una posición donde ya está almacenado un elemento, un hash abierto resuelve esta colisión encadenando el elemento a guardar a continuación del anterior. Es decir, cada posición del hash funciona como un balde o una cadena donde se guardan múltiples claves. Se requiere otro algoritmo de búsqueda para encontrar el elemento en cada posición.

Este tipo de direccionamiento se llama cerrado. Es decir, se garantiza que la la posición obtenida del hasheo es donde se encontrará el elemento (de existir).

2.1.1. Inserción

Para insertar una clave, se hashea y si la posición de llegada está vacía, se llena con la clave a insertar. Sino se sigue buscando hasta encontrar una posición vacía (y verificando que la clave no se encuentre en el hash).

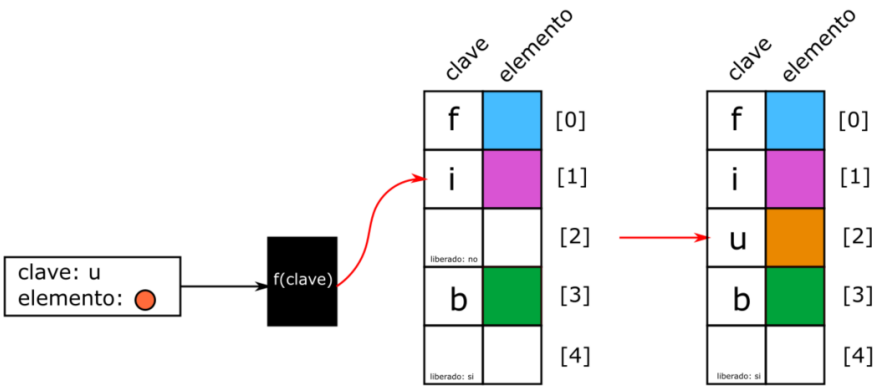


Figura 2: Hash abierto inserción.

2.1.2. Búsqueda

Para buscar una clave, se hashea y se verifica si coincide con la clave de la posición encontrada. Si no es así, se sigue avanzando por el vector hasta encontrar la clave o una posición vacía. Si la posición vacía corresponde a una clave anteriormente eliminada (indicada mediante alguna variable de la posición), se sigue buscando. Si la posición vacía jamás había sido llenada, entonces se deja de buscar: la clave no se encuentra en el hash.

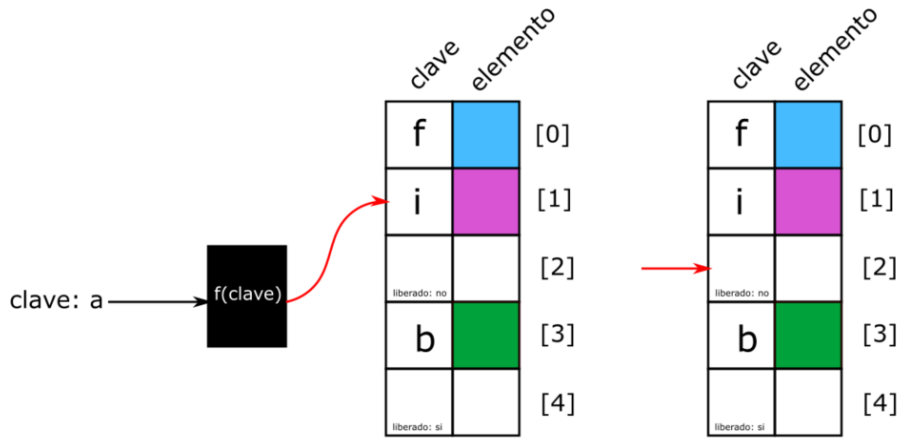


Figura 3: Hash abierto búsqueda.

2.1.3. Eliminación

Encontrada la clave, se elimina y se indica mediante la variable de control que la posición fue vaciada, lo cual es necesario para la operación de búsqueda.

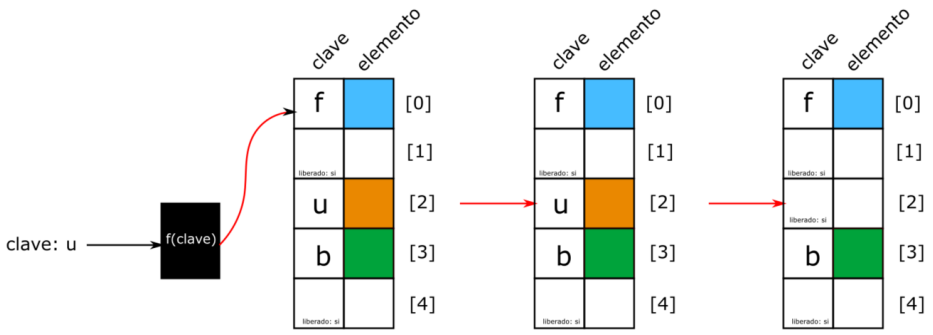


Figura 4: Hash abierto eliminación.

2.2. Hash cerrado

En el caso de un hash cerrado, cada posición de la tabla puede almacenar únicamente un elemento. Cuando se produce una colisión, se guarda o busca el elemento en la siguiente posición del vector hasta encontrar el elemento, o encontrar un lugar vacío donde guardarlo.

Este tipo de direccionamiento se llama abierto. Es decir, el elemento puede encontrarse en otra (idealmente, cercana) posición de la tabla.

2.2.1. Inserción

Para insertar se hashea la clave, y en la posición dada del vector, se agrega el elemento a la lista.

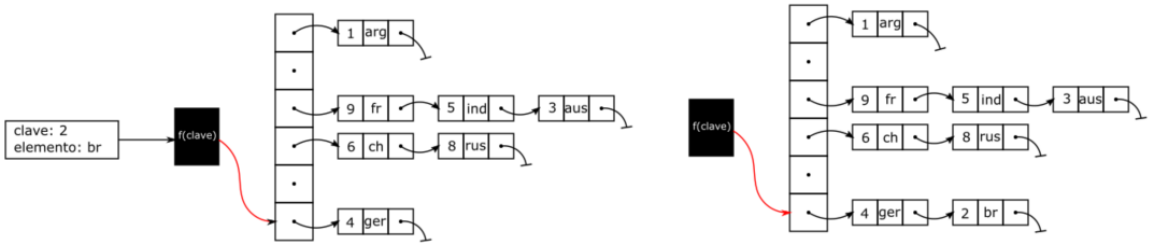


Figura 5: Hash cerrado inserción.

2.2.2. Búsqueda

Para buscar una clave, se hashea y se busca la clave en la lista en la posición dada del vector.

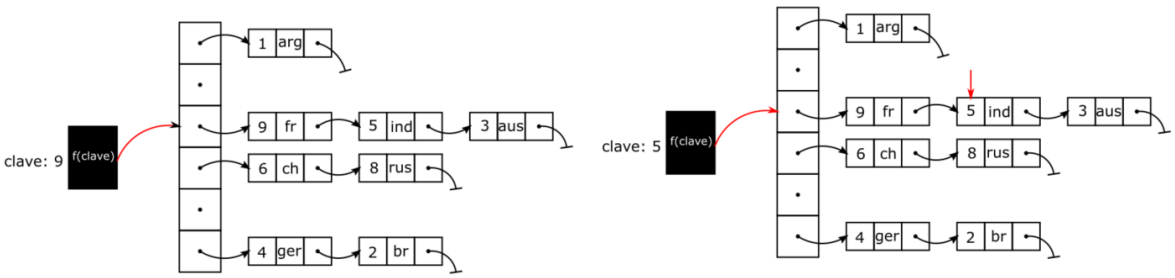


Figura 6: Hash cerrado búsqueda.

2.2.3. Eliminación

Encontrada la clave, se elimina el elemento, la clave y el nodo.

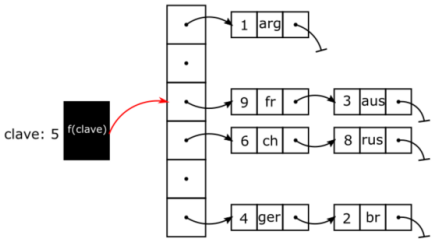


Figura 7: Hash cerrado eliminación.

3. Implementación y decisiones de diseño

En un primer momento se implementó el TDA con nodos simplemente enlazados, pero luego de varios intentos fallidos en la entrega por "timeout", se sospechó que las operaciones de búsqueda e inserción no eran lo suficientemente eficientes.

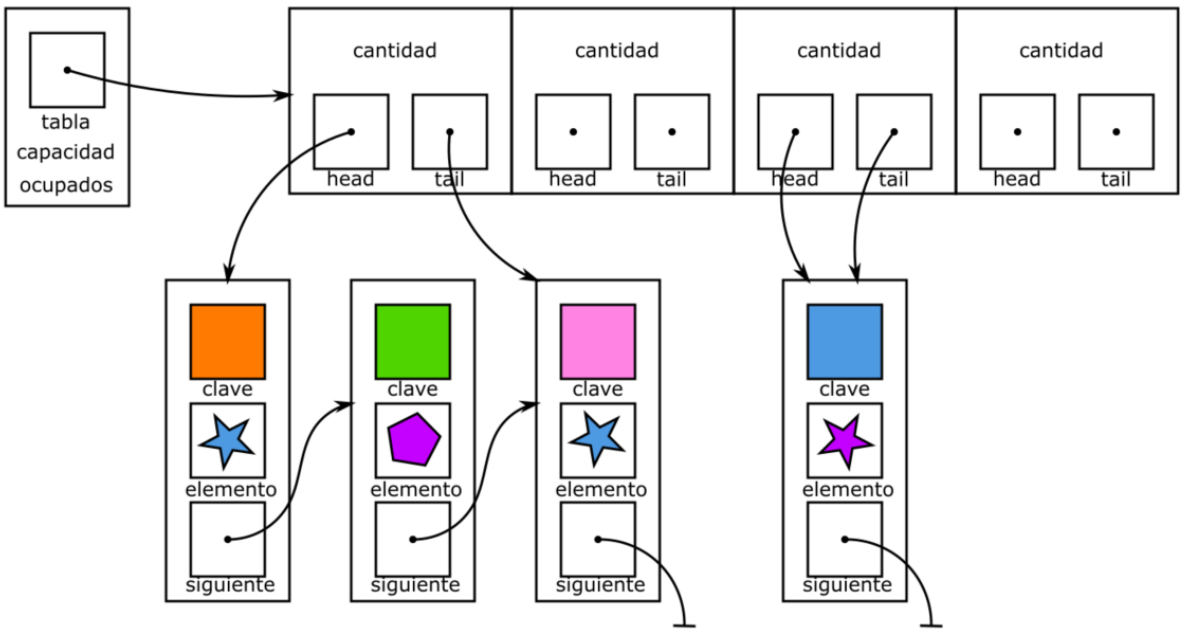


Figura 8: Estructura de la implementación.

Por este motivo el TDA se implementó con listas enlazadas. Es decir, cuando es creado el hash, la tabla consiste en un vector donde cada posición es un `struct lista`, que tiene como campos punteros a los nodos inicial y final de la lista (`head` y `tail`), y un contador de los elementos en dicha lista. Cada "nodo", llamado `struct entrada` consiste en un puntero a la clave, al elemento a guardar y a la siguiente entrada (o a NULL si fuese el último).

Con esta implementación se obtuvieron mejor resultados en cuanto a eficiencia. El razonamiento es que al tener un puntero al último elemento de la lista y el contador de los elementos, algunas operaciones puede realizarse de forma más eficiente.

Se utilizaron varias funciones auxiliares para manejar la inserción, eliminación, búsqueda e iteración en la lista, que son muy similares a las ya implementadas en el TDA lista.

Como criterio de rehasheo se planteó un factor de carga mayor o igual a 0,66; es decir que la cantidad de elementos en la tabla nunca supera el 66 % de la capacidad total de la tabla. Podría plantearse un criterio que considere un largo máximo de las listas, pero esto no se puso en práctica.