UNIVERSIDAD MAYOR, REAL Y PONTIFICA DE SAN FRANCISCO XAVIER DE CHUQUISACA

FACULTAD DE TECNOLOGIA

ESTUDIANTE:

- Amachuy Rodriguez Roberts

CARRERA:

- Ing. en Ciencias de la Computación

MATERIA:

- Estructura de Datos

N° DE PRACTICA:

- 8

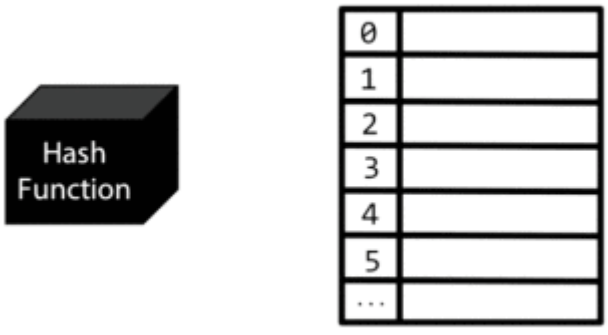
TABLAS HASH

Concepto: Estructura de datos no lineal cuyo propósito final se centra en llevar a cabo las acciones básicas en el menor tiempo posible, mejorando las cotas de rendimiento respecto a un gran número de estructuras.

Objetivos:

- Justificar la necesidad de las tablas hash.
- Comprender el funcionamiento de las principales operaciones de una tabla hash y su coste asociado.
- Conocer las principales características de una buena función hash.
- Entender el problema de las colisiones y los mecanismos para su resolución.

Definición: Las tablas hash son estructuras de datos que se utilizan para almacenar un número elevado de datos sobre los que se necesitan operaciones de búsqueda e inserción muy eficientes.
En palabras simples una tabla hash es solo la unión de un array(vector) y una función que llamamos Función Hash.



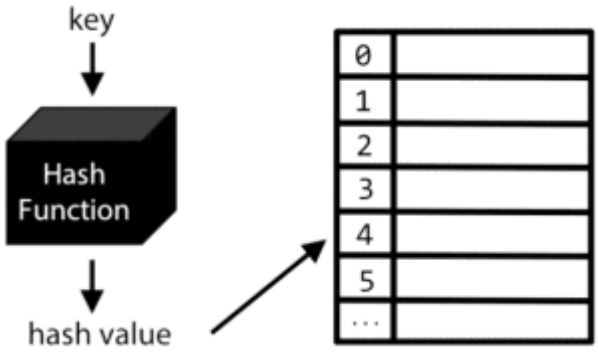
Podemos entonces decir que una Tabla Hash se forma por 3 elementos:

- **Una tabla** de un tamaño razonable para almacenar los pares(clave, valor).
- **Una función "hash"** que recibe la clave y devuelve un índice para acceder a una posición de la tabla.

- **Un procedimiento para tratar los casos** en los que la función anterior devuelve el mismo índice para dos claves distintas. Esta situación se conoce con el nombre de **colisión**.

Una Tabla hash almacena un conjunto de pares ”clave**, valor**”.

- **Dato/Clave/Key:** Es el parámetro de entrada, es el “dato que queremos guardar”
- **Valor/Indice/HashValue/Code:** Es un número, un entero positivo que nos indica donde debemos guardar nuestro “dato” dentro de la clave.



Función Hash

Esto es la clave porque nos indica donde hay que guardar nuestro dato y también donde es que debemos buscarlo.

Esta función **SIEMPRE** regresa lo mismo: **Un entero positivo.**

Usaremos ese número que nos regresa para saber en qué índice de un array vamos a guardar la información.

Debido a esto es muy importante que esta se comporte de maneta consistente.

- Tiene que minimizar las posibilidades de que ocurra una colisión como de lugar.
- Usará toda la información que sea proporcionada para maximizar la cantidad de códigos hash.
- Es determinista, es decir si le das una entrada x, siempre, tendrá que regresarte el mismo número como resultado.
- Distribuirá los valores de manera equitativa (ósea que cosas como “gatos” y “gata” acaben en lugares muy diferentes de la tabla).

- Tiene que ser capaz de generar índices muy diferentes para datos muy parecidos.
- Usar una función “sencilla” no hagas 50000 operaciones para generar cada índice.

Tamaño de la Tabla

El tamaño de una tabla hash es el otro parámetro que debe ser ajustado con cierto cuidado.

Por ejemplo, si el tamaño es demasiado pequeño, digamos 1, de nuevo la tabla se comporta como una lista encadenada.

En cambio, si el tamaño es enorme, el malgasto de memoria es evidente, puesto que habrá un elevado número de posiciones cuya lista de colisiones esta vacía.

Colisiones

Una colisión se produce cuando objetos diferentes dan lugar al mismo “valor” hash.

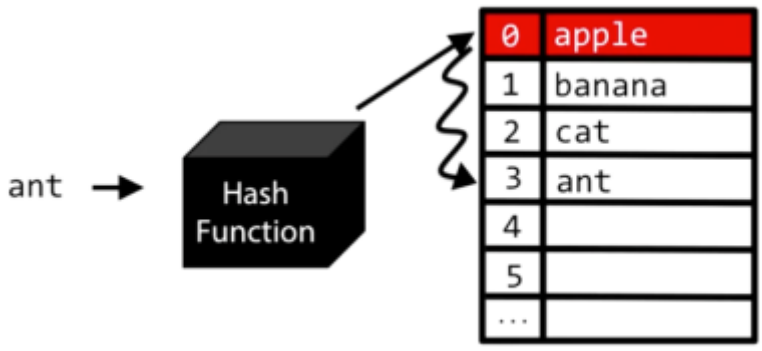
Es complicado evitar las colisiones: La función hash se debe calcular en tiempo constante y controlar las casillas ocupadas/libres implica un sobrecoste.

Pero hay dos formas de lidiar con esto:

Team Linear Probing

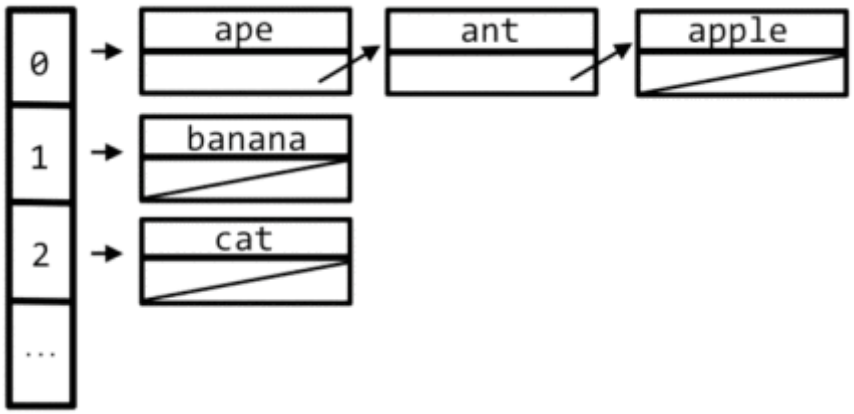
La solución de esta estrategia es simple: Si está ocupado ese lugar simplemente busca el más próximo que este libre.

Es decir, si un valor ya está ocupado buscar el valor próximo que este libre, para guardar el dato.



Team Separate Chaining

Esta estrategia propone que el array donde guardemos la información no sea más que un array de punteros hacia listas enlazadas.



Diferentes posiciones de la tabla pueden tener listas de diferente longitud o incluso vacías.

Conclusiones

- Las Tablas Hash permiten que el coste medio de las operaciones insertar, buscar y eliminar se constante.
- Se debe elegir correctamente la función hash:
  Debe ser fácilmente calculable, sin costosas operaciones.
  Debe tener una buena distribución de valores entre todas las componentes de la tabla.

BIBLIOGRAFIA

https://www.ecured.cu/Tabla_hash

http://bitybyte.github.io/Hashtables/

http://www.it.uc3m.es/pbasanta/asng/course_notes/ch07.html

https://youtu.be/LluB6jU-SwY

https://youtu.be/oWd0gpxQDQ4

https://youtu.be/9tZsDJ3JBUA

https://youtu.be/EMwhLbKMPYU