Estructuras de Datos

Estructura de Datos Tabla Hash

Prof. Luis Garreta

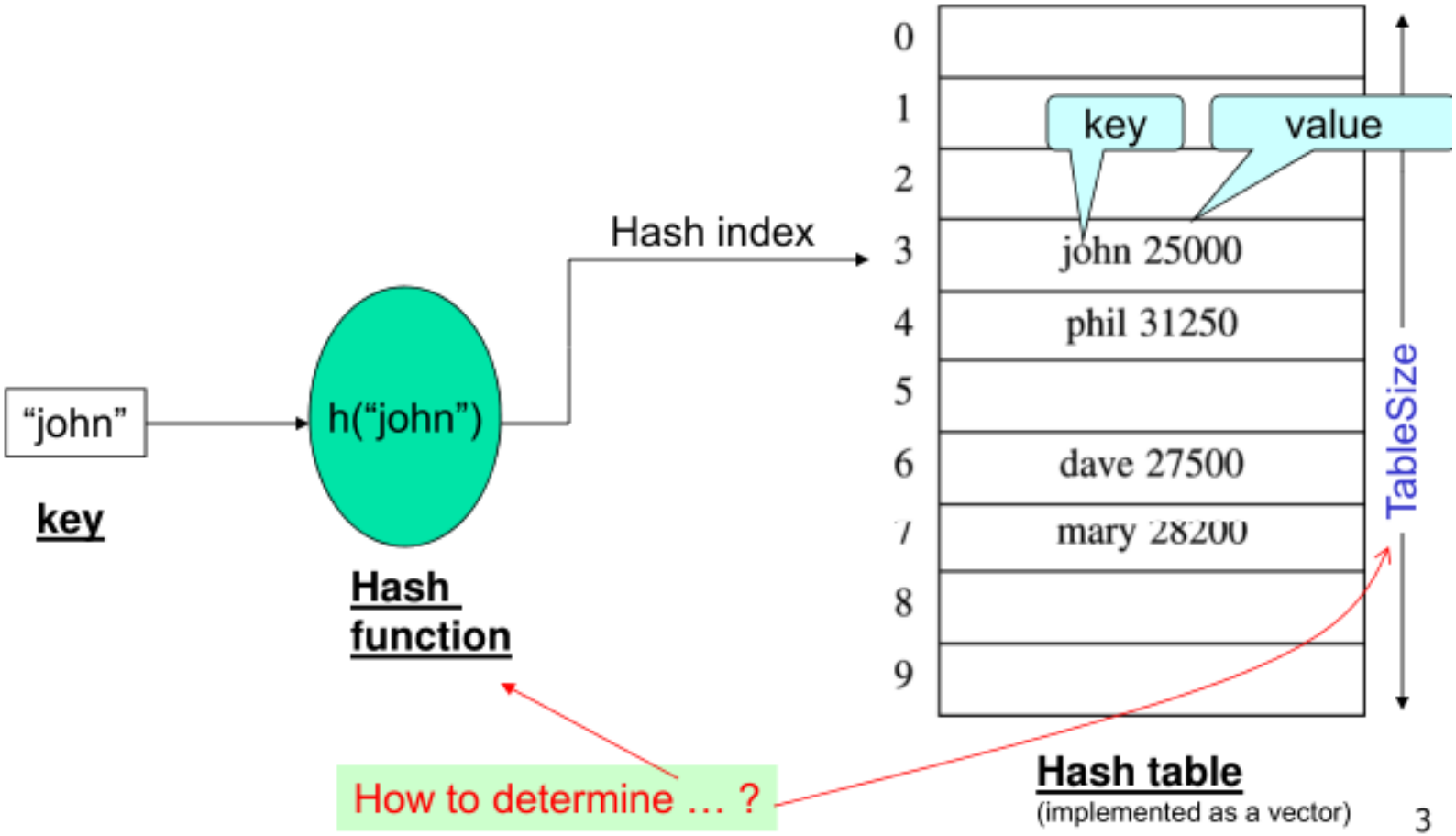
Ingeniería de Sistemas y Computación
Pontificia Universidad Javeriana – Cali

7 de noviembre de 2017

Introducción

- Proposito:
  - Las tablas hash soportan la inserción, eliminación y búsqueda en tiempo constante promedio (O(1)).
  - Se asume que el orden de los elementos es irrelevante.
  - Las tablas hash NO son útiles si se necesita mantener una relación de orden entre los elementos.
- Existe un función Hash
  - Hash ("clave") ==> valor entero
- TAD Tabla Hash:
  - Insertar, Buscar, Eliminar.

Componentes Principales Tabla Hash



Operaciones

- Insertar:
  - Tabla [fh("john")] = <"john, 25000>;
- Eliminar:
  - Tabla [fh("john")] = NULL;
- Busqueda:
  - Tabla [fh("john")] retorna el elemento "hashed" por "john".

Colisiones

Qué pasa si fh ("john") == fh ("joe")

Factores que afectan el diseño de la Tabla Hash

- La función hash
- El tamaño de la tabla
  - Generalmente asignado al principio
- El esquema de manejo de las colisiones

Función Hash

- Una función hash mapea (traslada) un elemento *llave (key)* en un índice válido en la tabla hash.
  - fh (key) => índice de la tabla hash
- La llave puede se de cualquier tipo (e.g. string, int, etc).
  - fh (string) => int
  - fh (int) => int
- Pero note que cualquier tipo puede convertirse en una forma de cadena equivalente:
  - 1234 => "1234"

Propiedades de la función hash

- La función hash *mapea* llaves a enteros
  - Restricción: Enteros resultantes entre [0..tamaño tabla -1]
- La función hash puede resultar en un traslado muchos-a-uno, causando colisiones:
  - Colisiones ocurren cuando una función hash traslada una o más llaves a un mismo índice del arreglo.
- Colisiones no pueden evitarse pero se pueden reducir usando una buena función hash.

Propiedades de la función hash

- Que reduzca el chance de colisiones:
  - Llaves diferente deberían transladarse a índices diferentes
  - Llaves se deberían distribuir uniformemente sobre toda la tabla.
- Debería ser *rápida* de calcular.

Funciónn Hash: Uso efectivo del tamaño de la tabla

- Una función hash simple (asumiendo claves tipo *int*):
  - fh (key) = key *modulo* Tamaño tabla;
- Para claves aleatorias, *fh* distribuye las llaves uniformemente sobre la table:
  - Qué pasa si el tamaño de la table = 100 y las llaves son TODAS múltiplos de 10?

Mejor si el tamaño de la tabla es un número primo

Diferentes formas de diseñar la función hash con claves tipo string

- Sumando el valor ASCII (0-255) para producir enteros:
  - E.g. "abcd" = 97+98+99+100 = 394
  - ==> fh ("abcd") == 394 % Tamaño de la tabla
- Problemas potenciales:
  - Anagramas:
    - fh ("abcd") == fh ("dbac")
  - Cadenas pequeñas pueden no usar toda la tabla.
  - Tiempo proporcional a la longitud de la cadena.

Técnicas para tratar con las colisiones

- Encadenamiento
- Direccionamiento abierto
- Doble hashing
- Etc.

Estrategia de encadenamiento

- Tabla hash **T** es un vector de listas enlazadas
  - Insertar elementos en el inicio *(head)* o al final *(tail)*
  - Clave **k** se guarda en la lista en **T (fh(k))**
- Ejemplo, Tamaño tabla = 10
  - fh (k) = k % 10
  - Insertar los 10 primeros cuadrados perfectos: {0, 1, 4, 9, 16, 25, 36, 49, 64, 81}