IIC-2133 — Estructuras de Datos y Algoritmos
Tablas de Hash

Jorge A. Baier

Departamento de Ciencia de la Computación
Pontificia Universidad Católica de Chile

Santiago, Chile

Objetivo y Aplicaciones

1. Muchas aplicaciones requieren “encontrar” un elemento rápidamente.
2. Las *Tablas de Hash* proveen operaciones para insertar y rescatar elementos rápidamente.
3. Esencial: una función (*de hash*) para mapear un universo de claves U a un conjunto $M = \{0, \ldots, m-1\}$.
4. m es el tamaño de la tabla de hash

Claves como números

- Es posible transformar cualquier clave a un número entero.
- Esto frecuentemente pasa por interpretar la clave como un número en alguna base.
- El número queda escrito de la forma

$$\sum_{i=0}^{n} = a_i b^i$$

- En ocasiones no combiene usar una base única, sino que interpretar la clave como una secuencia de bits.
- Ejercicio: escriba una función C que transforme una patente chilena (dos letras sequidas por dos letras o dos números, seguida por dos números) a un número entero.

El método de la división

1. El tamaño de la tabla de hash es frecuentemente limitado.
2. Una posibilidad es definir $h(k) = k \mod m$
3. ¿Qué valores son buenos (y malos) para m?

El método de la multiplicación

1. Sea A tal que $0 < A < 1$.
2. Se extrae la parte fracional de kA y se multiplica por m. En otras palabras:
$$\lfloor m(kA - \lfloor kA \rfloor) \rfloor$$
3. Ventaja: no es necesario preocuparse de m.

Colisiones

- Se producen cuando dos claves distintas k_1 y k_2 son tales que $h(k_1) = h(k_2)$.

Colisiones

- Se producen cuando dos claves distintas k_1 y k_2 son tales que $h(k_1) = h(k_2)$.

- Hay dos formas de resolver colisiones:
  - **Via encadenamiento**: la tabla de hash es un arreglo A de listas. El dato de clave k se almacena $A[h(k)]$.
  - **Direccionamiento abierto**: la tabla contiene referencia a datos. Si se desea insertar k, pero $A[h(k)]$ está lleno, se busca otra posición.

Hashing via Encadenamiento

- Las operaciones son INSERT, SEARCH, DELETE.
- Pseudocódigo: pizarra.

Hashing via Encadenamiento

- Las operaciones son INSERT, SEARCH, DELETE.
- Pseudocódigo: pizarra.

Definition

El *factor de carga* de una tabla de hash se define por $\alpha = n/m$, donde n es el número de datos y m el tamaño de la tabla.

Teorema: Todas las operaciones en hashing via encadenamiento es en promedio $\Theta(1+\alpha)$, si h distribuye claves en forma uniforme.

Demostración: pizarra.

Hashing Universal

- Una familia de funciones de hash $\mathcal{H} = \{h_1, \ldots, h_m\}$ con dominio en U y recorrido en $\{0, \ldots, m-1\}$ se dice *universal* ssi para todo $x, y \in U$ tal que $x \neq y$:

$$Pr_{h \in \mathcal{H}}[h(x) = h(y)] \leq \frac{1}{m}$$

Hashing Universal

- Una familia de funciones de hash $\mathcal{H} = \{h_1, \ldots, h_m\}$ con dominio en U y recorrido en $\{0, \ldots, m-1\}$ se dice *universal* ssi para todo $x, y \in U$ tal que $x \neq y$:

$$Pr_{h \in \mathcal{H}}[h(x) = h(y)] \leq \frac{1}{m}$$

- Si definimos $h_{a,b}(x) = ((ax + b) \mod p) \mod m$, con p un primo mayor que cada número en U, además: $p \geq m$ y $a, b \in \mathbf{Z}_p$

Teorema: La familia de funciones

$$\mathcal{H} = \{h_{a,b} : a \in \mathbf{Z}_p^*, b \in \mathbf{Z}_p\},$$

donde $\mathbf{Z}_p^* = \{1, \ldots, p-1\}\}$ es *universal*. (Demostración, pizarra.)

Direccionamiento Abierto (*Open Adressing*)

- La tabla no contiene listas sino que una referencia al dato.
- Al buscar/eliminar/insertar un elemento de clave k se busca en la tabla a los elementos

$$h(k,0), h(k,1), \ldots, h(k,m-1)$$

- Distintos tipos de "*probing*"
  - lineal: $h(k,i) = h'(k) + i$ (problema: clustering)

Direccionamiento Abierto (*Open Adressing*)

- La tabla no contiene listas sino que una referencia al dato.
- Al buscar/eliminar/insertar un elemento de clave k se busca en la tabla a los elementos

$$h(k,0), h(k,1), \ldots, h(k,m-1)$$

- Distintos tipos de "*probing*"
  - lineal: $h(k,i) = h'(k) + i$ (problema: clustering)
  - cuadrático: $h(k,i) = h'(k) + (-1)^{i+1}\lfloor \frac{i+1}{2} \rfloor^2$.

Direccionamiento Abierto (*Open Adressing*)

- La tabla no contiene listas sino que una referencia al dato.
- Al buscar/eliminar/insertar un elemento de clave k se busca en la tabla a los elementos

$$h(k,0), h(k,1), \ldots, h(k,m-1)$$

- Distintos tipos de "*probing*"
  - lineal: $h(k,i) = h'(k) + i$ (problema: clustering)
  - cuadrático: $h(k,i) = h'(k) + (-1)^{i+1} \lfloor \frac{i+1}{2} \rfloor^2$.
  - hashing doble: $(h_1(k) + ih_2(k)) \mod m$

Rendimiento de Direccionamiento Abierto

Teorema: El número esperado de intentos en una búsqueda no exitosa para una tabla de factor de carga $\alpha = n/m$ es a lo más $1/(1-\alpha)$, bajo el supuesto de uniformidad.

Rendimiento de Direccionamiento Abierto

Teorema: El número esperado de intentos en una búsqueda no exitosa para una tabla de factor de carga $\alpha = n/m$ es a lo más $1/(1-\alpha)$, bajo el supuesto de uniformidad.

Demostración: Se parte por definir la variable aleatoria

$$X = \text{número de intentos fallidos es exactamente igual a } i$$

y se calcula $E[X]$. Resto en la pizarra...

Rendimiento de Direccionamiento Abierto

Teorema: El número esperado de intentos en una búsqueda no exitosa para una tabla de factor de carga $\alpha = n/m$ es a lo más $1/(1-\alpha)$, bajo el supuesto de uniformidad.

Demostración: Se parte por definir la variable aleatoria

$$X = \text{número de intentos fallidos es exactamente igual a } i$$

y se calcula $E[X]$. Resto en la pizarra...

Corolario: El número de intentos en una inserción en una tabla con factor de carga α requiere a lo más $1/(1-\alpha)$ intentos en promedio, suponiendo uniformidad.

Rendimiento de Direccionamiento Abierto (2/2)

Teorema: El número esperado de intentos en una búsqueda exitosa para una tabla con factor de carga $\alpha < 1$ es a lo más $\frac{1}{\alpha} \ln \frac{1}{1-\alpha}$.

Rendimiento de Direccionamiento Abierto (2/2)

Teorema: El número esperado de intentos en una búsqueda exitosa para una tabla con factor de carga $\alpha < 1$ es a lo más $\frac{1}{\alpha} \ln \frac{1}{1-\alpha}$.

Demostración: Usamos el hecho de que la inserción de la i-ésima clave toma $1/(1 - i/m)$ intentos y promediamos sobre las n primeras inserciones.