

Diccionarios

Tablas de Dispersión (Hash)

Verónica E. Arriola-Rios

Facultad de Ciencias, UNAM

19 de octubre de 2022



Introducción

- 1 Introducción
- 2 Función de dispersión
- 3 Implementaciones

Temas

- 1 Introducción
 - Definición
 - Complejidades
 - Llaves

Definición

Definición (Diccionarios)

Un *diccionario* es un mapeo de un conjunto K de *llaves* a una colección V de *valores*.

- Se le considera una generalización de los **arreglos**.
- Una *función de dispersión* mapea llaves a posiciones en el arreglo.

Diccionario

Código: Dispersión

```

1 hash("José") -> 4
2 hash("Karla") -> 2
3 hash("Adalberto") -> 5
4 hash("Lucero") -> 0
5 hash("Pedro") -> 7
6 hash("Pablo") -> 9

```

0	Lucero	→	55 1278 9565	Tres Soles # 28
1				
2	Karla	→	44 5895 6235	Pingüinos # 15
3				
4	José	→	55 5896 2394	El Dorado # 3
5	Adalberto	→	55 9867 1425	Potros # 27
6				
7	Pedro	→	44 1658 9714	Amanecer # 2-102
8				
9	Pablo	→	35 4856 9874	Atardecer # 24

Temas

- 1 Introducción
 - Definición
 - Complejidades
 - Llaves

Complejidades

- Su característica esencial es la complejidad de estas operaciones:
 - Insertar $\rightarrow O(1)$
 - Encontrar $\rightarrow O(1)$
 - La eliminación puede ser costosa.
- Si conocemos a priori los datos que serán almacenados podemos garantizar que en el peor caso se cumplan las condiciones anteriores, si no, sólo en el caso promedio.
- Cumplir con estas cotas depende en gran medida de la complejidad de la **función de dispersión**.

Temas

- 1 Introducción
 - Definición
 - Complejidades
 - Llaves

Llaves

- K es el conjunto de valores posibles para las llaves.
- Las llaves se almacenan en un arreglo.
- La posición de la llave en el arreglo está dada por la función $h(x)$, con $x \in K$, llamada *función de dispersión* (o hash), que determina la posición de x , por el valor de x .
- En el caso general $|K|$ es grande o ilimitado. Por ejemplo:
 - ① Almacenar números de 32 bits $\Rightarrow |K| \approx 2^{32}$
 - ② Las cadenas de caracteres de tamaño arbitrario.
- Estimamos que el número de elementos **reales** a almacenar será considerablemente menor que $|K|$, i.e. $n \ll |K|$, de aquí que se necesitará un arreglo de tamaño M , con $n < M \ll |K|$.

Función de dispersión

- 1 Introducción
- 2 Función de dispersión
- 3 Implementaciones

Temas

2 Función de dispersión

- Definiciones
- Características
- Ejemplos

Función de dispersión

- 1 En general, una función de dispersión es un función $h : \mathbb{N} \rightarrow \mathbb{N}_{2^k}$ donde $\mathbb{N}_{2^k} = \{n \in \mathbb{N} | n < 2^k\}$, $k > 0$ **fija** y se dice que h tiene tamaño k . (Peláez 2018)
- 2 Hay una *colisión en la función* si para dos llaves $x \neq y$ tenemos que $h(x) = h(y)$.
- 3 La complejidad de su cálculo puede ser $\mathcal{O}(n)$ donde n es la longitud del natural a mapear, frecuentemente una representación en n **bytes** del objeto a dispersar.
- 4 Para valores semejantes x y y , $h(x)$ debe ser lejano a $h(y)$ provocando, efectivamente, una **dispersión** de las entradas.

Otras aplicaciones

Las funciones de dispersión tienen aplicaciones en criptografía. Por ejemplo:

- Los sistemas UNIX almacenan el código de dispersión de las contraseñas, en lugar de las contraseñas mismas.
- Cuando el usuario ingresa su contraseña se genera el código y se compara con el almacenado.
- En principio, es posible evitar colisiones porque el código de dispersión puede ocupar más bytes que la contraseña original.

Función de dispersión de búsqueda

- Una *función de dispersión de búsqueda* tiene requerimientos más relajados en cuanto a la ocurrencia de colisiones, pues no hay riesgos de seguridad involucrados.
- Es más estricta en cuanto a la complejidad en tiempo del cálculo, pues se espera que sea $\mathcal{O}(1)$.
- Para los diccionarios necesitamos una función de dispersión de búsqueda, donde:

$$h : K \rightarrow \{0, \dots, M - 1\} \quad (1)$$

- Dado que usualmente $|K| \gg M$, h por fuerza provocará *colisiones en el diccionario*, es decir, valores de K que son mapeados a la misma posición.
- En adelante, por *función de dispersión*, entenderemos una *función de dispersión de búsqueda*.

Temas

2 Función de dispersión

- Definiciones
- Características
- Ejemplos

Características de una buena función de dispersión de búsqueda

En resumen:

- ❶ Evita colisiones.
- ❷ Tiende a dispersar las llaves de manera uniforme.
- ❸ Fácil de calcular (en términos de complejidad computacional, idealmente es $\mathcal{O}(1)$).

A continuación se explican en detalle.

Evita colisiones

Dado un conjunto con $n < M$ llaves.

- Idealmente, para $k = \{k_1, \dots, k_n\}$ el conjunto de valores de dispersión $\{h(k_1), \dots, h(k_n)\}$ no contiene duplicados.

Tiende a dispersar las llaves de manera uniforme

- Sea p_i la probabilidad de que la función hash $h(x) = i$.
- Sea k_i el conjunto de llaves que se mapean al valor i , es decir $k_i = \{k \in K | h(k) = i\}$.
- Una función que distribuye las llaves uniformemente tiene la propiedad de que para $0 \leq i < M$, $p_i = 1/M$, es decir, los valores hash son uniformemente distribuidos.
- En ausencia de cualquier información *asumimos que las llaves son equiprobables*. Si este es el caso, los requerimientos para distribuir uniformemente las llaves implican que $|k_i| = |K|/M$, es decir, un número igual de llaves deberían mapearse en cada posición del arreglo.

Fácil de calcular

- No significa que sea fácil de calcular para una persona.
- No significa que sea fácil diseñar la función.
- Significa que **el tiempo de ejecución** de la función hash es $\mathcal{O}(1)$.

Temas

2 Función de dispersión

- Definiciones
- Características
- Ejemplos

Ejemplo: operación módulo

- Sea k cualquier tipo de llave:

$$h = g \circ f$$

$$f : k \rightarrow \mathbb{Z}^+$$

$$g : \mathbb{Z}_0^+ \rightarrow \{0, \dots, M-1\}$$

$$f(k) = x$$

- Método por división:

$$g(x) = |x| \% M$$

Ejemplo: cadenas

- Un algoritmo optimizado para dispersar cadenas es la función DJB2, creada por Daniel J. Bernstein.
- Éste jugará el papel de f en la diapositiva anterior.

Código: Función en el lenguaje C

```
1 unsigned long hash(unsigned char *str)
2 {
3     unsigned long hash = 5381;
4     int c;
5     while (c = *str++)
6         hash = ((hash << 5) + hash) + c; /* hash * 33 + c */
7     return hash;
8 }
```

Fuentes:

- <http://www.cse.yorku.ca/~oz/hash.html>
- <https://www.programandoamedianoche.com/2008/12/tablas-de-dispersion/>

Implementaciones

- 1 Introducción
- 2 Función de dispersión
- 3 Implementaciones**

Temas

- 3 Implementaciones
 - Dispersión abierta
 - Dispersión cerrada
 - Implementaciones alternativas

Arreglo de listas

- Se habla de *dispersión abierta* (*open hashing* o *separate chaining*) cuando los valores no serán almacenados dentro del arreglo correspondiente a la tabla, si no en estructuras adjuntas.
- El diccionario se programa como un arreglo de estructuras, donde el índice de cada casilla corresponde al código de dispersión de la llave y todos los objetos cuya llave sea mapeada a ese código son almacenados en la estructura.
- Ejemplos de estructuras para los valores son:
 - Lista de pares $\langle k, v \rangle$.
 - Árboles.
 - Arreglos ordenados.

<https://stackoverflow.com/questions/9124331/meaning-of-open-hashing-and-closed-hashing>

Implementación con arreglo de listas

```
1 hash("José") -> 4
2 hash("Karla") -> 2
3 hash("Adalberto") -> 2
4 hash("Lucero") -> 0
5 hash("Pedro") -> 0
```

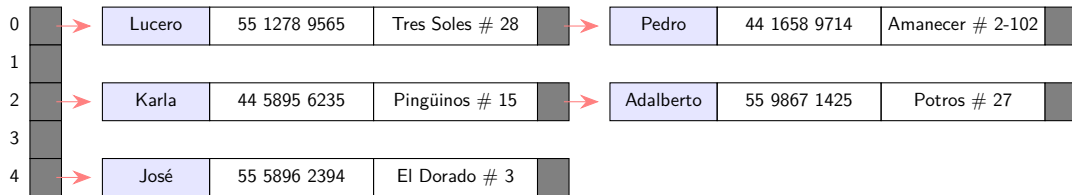


Figura: Implementación del diccionario

Terminología

- *Densidad de llaves*. Cociente entre el número de llaves en uso n y el número total de llaves posibles $|K|$.
- *Factor de carga*. Número de elementos almacenados entre número de registros disponibles.

$$\lambda = \frac{n}{M}$$
$$\lambda = \frac{1}{M} \sum_{i=0}^{M-1} n_i = \frac{n}{M}$$

Donde n es el número total de elementos almacenados y n_i es la i -ésima lista. Cuando éste esté arriba de 0.75 se puede decidir incrementar el tamaño del arreglo subyacente, con la penalización correspondiente en tiempo.

<https://www.geeksforgeeks.org/load-factor-and-rehashing/>

Complejidad de las operaciones

Peor caso:

- Inserción $O(1)$ (si se inserta a la cabeza de la lista, por ejemplo)
- Borrado $O(\mu)$
- Búsqueda $O(\mu)$

donde μ es el máximo número de colisiones por llave.

Temas

- 3 Implementaciones
 - Dispersión abierta
 - Dispersión cerrada
 - Implementaciones alternativas

Dispersión cerrada

- Se habla de *dispersión cerrada* (*closed hashing* u *open addressing*) cuando toda la información es almacenada dentro del arreglo correspondiente a la tabla de dispersión.
- La *función de dispersión* indica la posición donde un elemento debería ser almacenado.
- Cuando ocurre una **colisión**, el elemento colisionado es almacenado en algún otro lugar del arreglo.
- La posición final del elemento queda determinada por el tipo de *direccionamiento* elegido para resolver las colisiones.

	Llave	Valor	Sig (opt)
0	din		nil
1	foo		2
2	bar		3
3	tmp		nil
4	baz		nil
5	kas		nil
6	jaz		7
7	zip		9
8	sox		nil
9	kk		nil

Direccionamiento abierto

- Hay que definir una secuencia de exploración para cada llave que, cuando se sigue, nos lleva a la posición final de la llave en cuestión.
- Es una secuencia de funciones:

$$\{h_0(x), h_1(x), \dots, h_{M-1}(x)\} \quad (2)$$

donde cada h_i es una función de dispersión $h_i : K \rightarrow \{0, 1, \dots, M-1\}$.

Secuencias de exploración

Las secuencias de exploración más comunes son de la forma

$$h_i(x) = (h(x) + c(i)) \% M \quad (3)$$

Donde i es el número de intento y la función $c(i)$ representa la *estrategia* de resolución de colisiones, que debe satisfacer:

- 1 $c(0) = 0$
- 2 El conjunto de valores $\{c(0) \% M, c(1) \% M, \dots, c(M-1) \% M\}$ **debe contener a todos los enteros entre 0 y $M-1$.**

Exploración lineal

En general tiene la forma:

$$c(i) = \alpha i + \beta \quad (4)$$

Por ejemplo:

$$c(i) = i$$

entonces la secuencia tiene la forma:

$$h_i(x) = (h(x) + i) \% M$$

explícitamente:

$$h_0(x) = h(x) \% M$$

$$h_1(x) = (h(x) + 1) \% M$$

$$h_2(x) = (h(x) + 2) \% M$$

Exploración cuadrática

En general:

$$c(i) = \alpha i^2 + \beta i + \gamma \quad (5)$$

Por ejemplo:

$$c(0) = 0$$

$$c(i) = i^2$$

explícitamente:

$$h_0(x) = h(x) \% M$$

$$h_1(x) = (h(x) + 1) \% M$$

$$h_2(x) = (h(x) + 4) \% M$$

$$h_3(x) = (h(x) + 9) \% M$$

Teorema

Cuando se usa exploración cuadrática en una tabla de tamaño M , donde M es un número primo, las primeras $\lfloor M/2 \rfloor$ exploraciones son distintas.

Dem. por reducción al absurdo.

Sean dos valores i, j tales que $0 \leq i \leq j < \lfloor M/2 \rfloor$

$$h_i(x) = h_j(x) \Rightarrow h(x) + c(i) = h(x) + c(j) (\%M)$$

$$\Rightarrow i^2 = j^2 (\%M)$$

$$i^2 - j^2 = 0$$

$$(i - j)(i + j) = 0$$

Pero

❶ $(i - j) \neq 0$

❷ $(i + j) \neq 0$

Dispersión doble (*double hash*)

Se usa una segunda función de dispersión para definir la estrategia de exploración:

$$h : K \rightarrow \{0, \dots, M - 1\} \quad (6)$$

$$h' : K \rightarrow \{1, \dots, M - 1\} \quad (7)$$

$$h_i(x) = (h(x) + ih'(x)) \% M \quad (8)$$

Sugerencias:

$$h'(x) = 1 \quad \text{mala idea :)}$$

$$h'(x) = i \quad \text{mala idea también :)}$$

$$h = g \circ f \quad \rightarrow f \text{ mapea llaves a enteros}$$

$$g(x) = x \% M$$

$$h' = g' \circ f$$

$$g'(x) = 1 + (x \% (M - 1))$$

Resumiendo

① Direcccionamiento abierto

$$\{h_0(x), h_1(x), \dots, h_{M-1}(x)\} \quad (9)$$

② Exploración lineal

$$h_i(x) = (h_0(x) + \alpha i + \beta) \% M \quad (10)$$

③ Exploración cuadrática

$$h_i(x) = (h_0(x) + \alpha i^2 + \beta i + \gamma) \% M \quad (11)$$

④ Doble hash

$$h_i(x) = (h(x) + i h'(x)) \% M \quad (12)$$



Temas

- 3 Implementaciones
 - Dispersión abierta
 - Dispersión cerrada
 - Implementaciones alternativas

Llaves ordenadas

- Una desventaja de utilizar funciones de dispersión y arreglos es que no se preserva ningún tipo de orden sobre las llaves:
- Para conservar un orden se puede utilizar un árbol ordeando balanceado para almacenar las llaves. Por ejemplo, Java usa rojinegros para su `TreeMap`. Esto cambia la complejidad de las operaciones a $\mathcal{O}(\log n)$.

Bibliografía I

-  *Notas del curso de Estructuras de Datos de la profesora Elisa Viso (2002).*
-  Peláez, Canek (2018). *Estructuras de datos con Java moderno*. Temas de Computación. Las prensas de ciencias.

Licencia

Creative Commons
Atribución-No Comercial-Compartir Igual

